BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisch s Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 137 226 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
26.09.2001 Bulletin 2001/39

(51) Int Cl.[7]: **H04L 12/56**

(21) Application number: **01000050.3**

(22) Date of filing: **09.03.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **22.03.2000 US 191375 P**

(71) Applicant: **Texas Instruments Incorporated**
Dallas, Texas 75251 (US)

(72) Inventor: **Welin, Andrew M.**
California93103, Santa Barbara (US)

(74) Representative: **Holt, Michael**
Texas Instruments Ltd.,
EPD MS/13,
800 Pavilion Drive
Northampton Business Park,
Northampton NN4 7YL (GB)

(54) Improved packet scheduling of real time information over a packet network

(57) One method of processing first and second received packets of real-time information includes computing for each of said received packets respective deadline intervals and ordering processing of the first and second received packets according to the respective deadline intervals. A single-chip integrated circuit has a processor circuit and embedded electronic instructions forming an egress packet control establishing an egress scheduling list structure and operations in the processor circuit that extract a packet deadline intervals DI, place packets in the egress scheduling list according to deadline intervals DI; and embed a decoder that decodes the packets according to a priority depending to their deadline intervals. Embedded electronic instructions establish an egress scheduling list structure and operations in the processor circuit that establish channel decoders on non-coincident frame boundaries and a packet engine to detect when a first packet has a first deadline and is currently in decode while a second packet is just-arriving and has a second deadline earlier than the first deadline. The packet engine establishes a determination whether both the second and first packets can be decoded ahead of their respective deadlines if the second packet were decoded preemptively, and if so, then preempts the processor circuit channel decoder structure to decode the second packet. Other processes, integrated circuits, chipsets, wireless telephones, PBXs, line cards, computers, routers, and networks for voice over packet, media over packet and real-time transmissions over packet are disclosed.

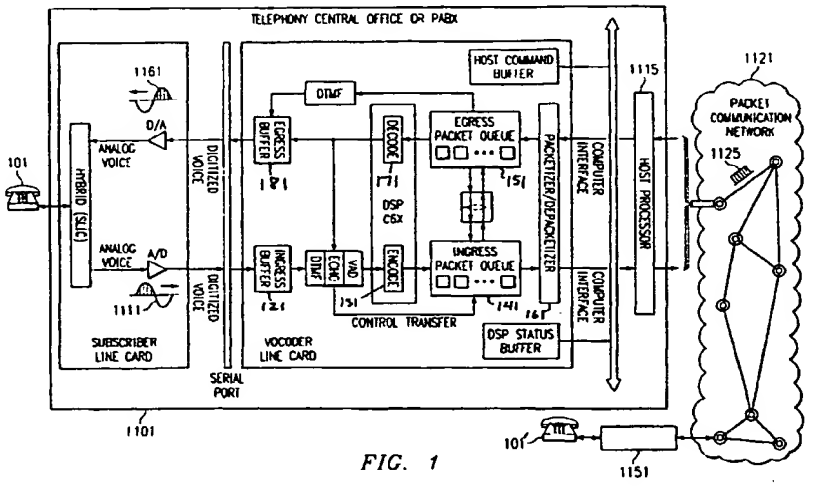


FIG. 1

EP 1 137 226 A2

Description

Technical Field of The Invention

[0001] The present invention relates to the fields of integrated circuits, networking, systems and processes for packet communications, and especially communication of real time information such as voice, audio, images, video and other real time information over packet.

Background Of The Invention

[0002] For every frame of compressed speech in a packet, loss of that packet means loss of each frame in that packet. There then arises the problem how to create 160 bits or more of lost compressed speech. Reduction of packet loss and late packet handling strategy are very important challenges in advancing VOP technology.

[0003] Telephony represents a duplex channel. In the case of packet telephony one side (the ingress side) receives voice or digitized voice (PCM data) and produces packets by using any of several compression processes. This ingress side is almost completely 'synchronous'. Voice is changed into frames. The size of the frames for a given data compression process is fixed.

[0004] On the other side (the egress side) of packet telephony the packets are converted to PCM frames, which (frames) are added to output buffers for each channel. The packets arrive at rate for which only the average if known. This average depends on the process used and thus on the frame size to be produced. The data from the output buffer is output at a constant rate. If not replenished in time, the data will run out.

SUMMARY OF THE INVENTION

[0005] In one form of the invention, a method of processing first and second received packets of real-time information includes computing for each of said received packets respective deadline intervals and ordering processing of the first and second received packets according to the respective deadline intervals.

[0006] In yet another form of the invention, a single-chip circular time differencing integrated circuit has a storage for values representative of the time of two events. An adder/subtractor coupled to the storage generates an electronic difference (delta) and delivers the difference value into the storage thereby resulting a sign bit (S) and a most significant bit (MSB) of the difference value (delta). Logic circuitry responds to the MSB and the sign bit S of the electronic difference (delta) and a predetermined value (TMAX), to drive the adder/subtractor to generate the circular time difference of the two events.

[0007] In still another form of the invention, a wireless telephone includes an antenna, a voice transducer, and at least one integrated circuit assembly coupling the voice transducer to the antenna, and providing voice over packet transmissions and embedded electronic instructions comprising an ingress/egress packet control that processes egress information and determines lowest first egress deadline interval DI and further executes an ingress process preempting the egress process when the value of lowest first egress deadline interval DI exceeds a predetermined amount.

[0008] Other forms of the invention encompass other processes, integrated circuits, chipsets, line cards and other computer add-in cards, information storage articles, systems, computers, gateways, routers, cellular or other wireless telephone handsets, wireless base stations, appliances, and packet networks, and other forms as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The present invention will now be further described, by way of example, with reference to the preferred and exemplary embodiments illustrated in the figures of the accompanying drawings in which:

FIG. 1 is a block diagram of an inventive process, integrated circuit, line card, system and packet communication network;

FIG. 2 is a partially pictorial, partially block diagram of various inventive computers, wireless telephones, PBXs, automotive systems, and networks, and FIG. 2 includes a magnified view of an inventive router implemented in networks;

FIG. 3 is a partially block, partially pictorial diagram of an inventive packet network enabled PBX serving telephones, fax/scanners, and wireless telephones for communication with the Public Switched Telephone Network (PSTN) and a packet data network;

FIG. 4 is a partially pictorial, partially block diagram of inventive wireless telephones with network access and improved for enhanced packet communications;

FIG. 5 is a block diagram of an inventive process for software and one or more inventive integrated circuits for enhanced packet communications;

FIG. 6 is a partially block, partially process diagram of inventive processes and one or more inventive integrated circuits for enhanced packet communications;

FIG. 7 is a diagram of an example type of packet for use with the inventive processes, integrated circuits and systems herein;

FIG. 8 is a diagram of inventive arrival, queuing, and processing operations wherein a horizontal axis represents time, and a vertical axis portrays various channels of a multi-channel packet communications system;

FIG. 9 is a mostly block, partially channels versus time, diagram of an inventive egress processing system that processes communications packets arriving from a network;

FIG. 10 is a diagram of arrival of packets in various communications channels having codecs operating on different frame lengths wherein a horizontal axis represents time, and a vertical axis portrays various channels of a multi-channel packet communications system;

FIG. 11 is a block diagram of an inventive embodiment of buffers and decoder(s) for improved packet communications;

FIG. 12 is a graph of Usefulness versus a parameter X wherein the graph depicts operations and advantages of various inventive embodiments;

FIG. 13 is a flow chart of inventive process steps based on packet deadlines;

FIG. 14 is a flow chart of inventive process steps for handling information in various types of packets;

FIGS. 15, 16, and 17 are diagrams of inventive and related data structures, records, and inventive updates processes for use in inventive processes, integrated circuits, and systems;

FIG. 18 is a block diagram of an inventive data structure for a queue having primary and secondary keys;

FIG. 19 is a block diagram of an inventive system emphasizing interrupt structures;

FIG. 20 is a chart of inventive embodiments according to interrupt or preemption policies wherein different embodiment types are represented on different rows, and handling of differently timed packets is grouped in columns according to time of arrival of a packet;

FIG. 21 is a comparative timing diagram of two categories of inventive processes-Same-Deadline processes and Staggered Deadline Processes, wherein an example of a Staggered Deadline process is spread vertically and horizontally in a lower part of FIG. 21;

FIG. 22 is a flowchart of an inventive Staggered Deadline Process;

FIG. 23 is a flowchart of another inventive Staggered Deadline Process;

FIG. 24 is a flowchart of yet another inventive Staggered Deadline Process;

FIG. 25 is a flowchart of an inventive Break Process in the process of FIG. 24;

FIG. 26 is a flowchart of an inventive process example of FIG. 20 Embodiment #5, upper left cell "Do Ingress First";

FIG. 27 is a block diagram of an alternative process to that of FIG. 14;

FIG. 28 is a comparative time diagram of various cases of inventive process operations of converting linear time differences to circular time differences, for use in computing Deadline Intervals in other inventive processes herein;

FIG. 29 is a flow chart of operations of in an inventive process for comparison with the time diagram of FIG. 28, wherein the process converts linear time differences to circular time differences;

FIG. 30 is partially block diagram of a register, and partially graphical illustration of the various cases of FIG. 28 wherein a variable of linear time difference Delta is used, and various operations of add and subtract (or neither) are employed;

FIG. 31 is a partially block, partially pictorial diagram of an integrated circuit chip improved as described herein;

FIG. 32 is a partially block, partially flowchart, diagram of operations corresponding to those of FIGS. 28-30 implemented in inventive hardware circuitry and process;

FIG. 33 is a flowchart of inventive process embodiment that sorts new information into a queue and utilizes the circular time difference process of FIG. 29;

FIG. 34 is a block diagram of an inventive system for image, video, speech, and audio improved packet communications;

FIG. 35 is a pictorial diagram of one embodiment of an inventive storage article of manufacture improved with physical variations establishing an inventive sequence queue process, and other inventive processes described herein;

FIG. 36 is a pictorial diagram of another embodiment of an inventive storage article of manufacture having a disk drive, control sytem and system computer add-in card improved with physical variations and software establishing one or more inventive processes described herein;

FIG. 37 is a block diagram of an inventive Internet appliance system improved with inventive processes and inventive integrated circuits as described herein; and

FIG. 38 is a flowchart of an inventive process for handling gaps or holes in buffer reserves of data, when such

gaps or holes occur in operations of inventive integrated circuits and systems as described herein.

[0010] In the Figures, corresponding numerals refer to corresponding parts and steps, except where the context indicates otherwise.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0011] FIG. 1 presents a view of packetized telephony. Packets 1125 (see also FIG. 7) comprise bit streams with headers and their payloads of one or more compressed "frames" of voice data. RTP (Real Time Transport Protocol) and RTCP (RTP Control Protocol) add time stamps and sequence numbers to the packets, augmenting the operations of the network protocol such as Internet Protocol. Each frame, depending on the compression process being applied, comprises 10, 20, 30 or 40 milliseconds of digitized voice. The telephony standard sampling rate converting voice signal into digitized voice data is usually 8kHz, although other rates are suitably used.

[0012] In FIG. 1 telephones 101 and 101' typify plural sources and destinations of voice signals. Telephone 101 (inputs and outputs) is indirectly connected to an Analog to Digital (ADC) and Digital to Analog (DAC) converters, which produce and receive digitized voice.

[0013] In FIGS. 1 and 2 an analog electrical signal 1111 is produced by a transducer (audio, light, pressure, temperature, or any other physical quantity, not shown) , whereupon it is sampled as indicated by vertical lines thereon. The samples are numbers so related in time so that, when they are sent by a computer 1101 via a transmission medium 1121. such as the Internet or other network, to a receiving computer 1151, they can be reconstructed as another analog signal 1161. Computers 1101 and 1151 are suitably constructed to handle many signals of the type of 1111 at the same time.

[0014] The computer 1101 creates frames of digitized data. It acts as an interface wherein the digital signal is broken into pages, or frames or buffers. An important event in computer processing is buffer interrupt. When a buffer is filled with data, then an interrupt is generated by the buffer and coupled to a processor 1115, and the processor is thereby signaled that it has a full buffer of data to process.

[0015] A similar event occurs on the output processing of the computer 1151. The computer 1151 creates the pages. In the case of the packet, the pages are created not out of the voice or other analog waveform 1111 directly but out of packets 1125 that are suitably decoded by computer 1151. The issue here is that at some point the computer 1151 may have created a page or buffered a data frame while a previous frame is being output. When the previous frame of data ends the parent page has to be ready or there will be a break in continuity of the output waveform 1161 being generated by computer 1151. Suppose a packet 1125 is late arriving at computer 1151. Then the break is usually just filled with silence (or gap in the real time output), or filled with noise, or filled with a copy of previously received data.

[0016] The Central Office on FIG. 1 is shown to include a Subscriber and Vocoder Linecards. Each linecard is designed to support multiple lines. Using a different phrase, they are designed to support multi-channel operations.

[0017] In FIG. 1, Host processor 1115 with DSP C6201 coupled thereto. The host 1115 has a handshake host control process provided therein to exchange information between the host and the source or destination connected to the host. The host controls what a given channel is doing.

[0018] In a three-part ingress process the host is a sender. The host 1115 in an ingress initiation process detects when a handset 101 is picked up and dials a destination, and then the host opens a channel and sends signaling packets indicative of initiation of the call. In an ingress communication process, the host then sends voice data packets to a destination. In an ingress termination process, the host detects whether the handset 101 has been put down, and then the host closes the channel by sending signaling packets indicative of termination of the call.

[0019] Also, the host responds to incoming calls with an egress initiation process, an egress communication process, and an egress termination process. In the egress initiation process, the host 1115 receives signaling packets from another computer indicative of initiation of a call to host 1115. The host 1115 interacts with the DSP so that, among other things, a channel number is assigned to the call being initiated and the decode process to be used has an process identifier stored into the egress channel record 1413 of FIGS. 14 and 15. In this way the channel record characterizes the channel for the host and the DSP to use, and structures the operations of the DSP relative to that channel until the channel is disconnected.

[0020] In the egress communication process, the host 1115 receives voice data packets from the other computer 1151 and decodes them using the process identified by the process identifier in the egress channel record 1413 for that channel over which the voice data packets are coming from the other computer. It is precisely in the egress communications process that improvements of some embodiments such as of FIG. 11 and 12 are provided, so as to make use of packets which would otherwise be effectively lost to a process hitherto.

[0021] In the egress termination process, the host 1115 in FIG. 1 receives signaling packets from the other computer indicative of termination of the call to host 1115. The host 1115 interacts with the DSP so that, among other things, the channel is closed and the egress channel record 1413 is suitably updated, released or closed.

[0022] 32 channel management system combined with a 32 channel decoder on the egress side. Note that the decoder is simply a program, and the computer has a set of, for example, five decoder programs implemented five corresponding decode processes of which one might be G.723. A given one of the decoder programs services all of the channels that call for its decode process in block 1413, channel by channel. Block 1413 determines which decoder is assigned to which channel. All 32 channels may use the same decoder. Or 5 channels might use decoder 1, 12 channels use decoder 2, 9 channels use decoder 3, 2 channels use decoder 4, and 4 channels use decoder 5, for another example.

[0023] Many but not all embodiments have a decoder as in FIG. 1 in the same device, system, or process at a given node of implementation. However, still other embodiments have the decoder physically located remotely, over a synchronous network or otherwise, from the packet shuffling or sorting process site of FIG. 2A. At the network level, the network 200 of FIG. 2 constitutes a system wherein each packet has its own hard deadline. Many packets are channeled into a node or router (e.g., 133) in the network 200 to be sent out in order of priority by a packet scheduling manager of FIG. 2A. If it is known when the speech begins, then it is known exactly when the packet is needed utilizing a channel delay database 1171. Then the system is extended and arranged to make the process all remote, wherein the router and packet scheduling manager of FIG. 2A is provided with a process that decides how to send out packets like 1125 from buffer 1163 to a destination many nodes away. Thus, a buffer selector 1175 coupled to an issuer 1165 decides when to send various packets SLOW and FAST from the router out to destination nodes. In this way, packets that would otherwise arrive too late at the destination decoder 171 are in fact advanced and issued by issuer block 1165 in the routing process so that they arrive in time for use at the decoder 171 at the end of the path.

[0024] FIG. 2 shows a network cloud 200 coupling computers 203 and 205. If one path from a source 203 is intermittent, then another path is made to be present so that packets can get to the destination 205. The source 203 inventively launches packets and any dependent packets one or more paths through network 200. In the Internet the path that a given packet will take cannot usually be predicted, and various packets will take different routes due to the fault-tolerant, multiple path nature of the Internet. A PC or workstation is provided at destination 205 to receive streams of data such as from intermediate nodes 231 and 233.

[0025] Further in FIG. 2, personal computer 203 has a microphone 261.1, a loudspeaker (and/or headphones or other audio transducer) 262.1, a keyboard (and/or mouse or other touch-sensitive input device) 263.1, a computer box 264.1 including one or more information storage devices 265.1 and a printed wiring board 266.1 with microprocessor (s), digital signal processor(s), volatile memory, peripheral chipset and peripherals. Associated with computer box 264.1 is a cathode ray tube monitor (and/or liquid crystal display, and/or digital light processor (DLP™ Technology from Texas Instruments Incorporated) and/or other display device and/or printer) 267.1 coupled to printed wiring board 266.1. Other peripherals (not shown) such as videoconferencing camera, digital still camera, optical scanner, electrocardiograph EKG, wire/power line cable/fiber networking interfaces, wireless networking interface and other devices now available or yet to be devised are also coupled to printed wiring board 266.1. A modem 268.1 is also coupled to printed wiring board 261.1. The modem is suitably V.90 voice-band modem, cable modem, DSL (digital subscriber line modem), ISDN (Integrated Services Digital Network) or other suitable modem. The modem 268.1 couples personal computer 203 to a packet network gateway computer 271 as well as to a public switched telephone network PSTN 285.

[0026] A similar description applies to various components associated with computer 205 of FIG. 2, and reference numerals with a suffix ".i" have like description of corresponding reference numerals already described in connection with personal computer 203. Also the suffix ".i" indicates that computer 205 is one of many computers coupled to packet network 200 and or via PSTN 285 to a gateway to network 200.

[0027] Further in FIG. 2, a cell phone 281 typifies numerous cell phones active in a cell of a cellular telephone base station 283. Cell phone 281 has an enclosure with a manual input (or touch pad or button pad or keyboard) 281.1, a microphone 281.4, an audio output transducer such as a loudspeaker 281.5, a visual interface 281.3 such as an LCD screen, and a wireless antenna 281.7. Inside of cellular telephone 281 is electronics coupled to the aforementioned components, and the electronics includes an analog section coupling the microphone 281.4 and speaker 281.5 to an integrated circuit assembly including TMS320C54xx and/or TMS320C55xx DSP from Texas Instruments Incorporated and a microcontroller such as an ARM (TM) chip licensed by Advanced RISC Machines. The microcontroller is also coupled to the manual input 281.1 and visual interface 281.3. Further, the microcontroller is coupled with the digital signal processor. A radio frequency RF section couples the other sections and chips to the antenna 281.7 for two-way and multi-way communications.

[0028] Base stations 283 and 287 are coupled to a public switched telephone network PSTN 285, which in turn is coupled to the packet network 200. Also, base stations 283 and 287 are respectively coupled to packet network 200 via gateways 291 and 293. In the cell served by base station 287, a cell phone 289 typifies numerous cell phones active in a cell service area of that base station 287.

[0029] A private branch exchange PBX 202 couples telephones 204 and 206 to PSTN 285. Suitably, PBX 202 is improved for path diversity communications as described herein. Another PBX 211 couples IP phones 213 and 215 to a node of packet network 200 as illustrated.

[0030] In FIG. 3, system components are arranged to provide gateway functions and combined with cellular phone base-station functions and PBX functions. A communication system 301 interfaces to a PSTN (public switched telephone network) 303, to a telephone 305 (and PBX private branch exchanged connected to many wired and cordless telephones, not illustrated), to a fax and/or scanner machine 307 and to cellular telephones 309. PSTN 303 is coupled via T1/E1 Framer 311 to a DSQ Switch 341. Telephone 305 and Fax 307 are coupled via a PCM Codec 321 to the DSQ Switch 341. Cellular telephones 309 are coupled via a wireless communications interface 331 to the DSQ Switch 341.

[0031] Further in FIG. 3, the DSQ switch 341 couples the various types of communications to a first port of a bank of one or more DSPs (digital signal processors, such as TI TMS320C6x or TMS320C54x DSPs) 351, 353, and so on to the Nth DSP 355 in the DSP bank. Each DSP suitably has associated memory 361, 363,...365 respectively provided as any suitable mix of volatile and nonvolatile memory selected by the skilled worker. The DSPs are connected via a second port of the bank to a bus 371 which couples them to a microcontroller 381 that has its own RAM memory 383 and flash nonvolatile memory 385. The microcontroller 381 communicates via a PHY, or Network Physical Interface 391, to packet data network 200 of FIG. 2.

[0032] In FIG. 3, various parts of the improvements described herein are suitably partitioned between the DSPs 351, 353,...355 and the microcontroller (MCU) 381 and stored on-chip and in the off-chip memories as desired. Various partitioning alternatives are contemplated. Also, the MCU is omitted in another embodiment (not shown) and the various software blocks are partitioned among execution units of one DSP or among multiple DSPs.

[0033] Software as disclosed herein is also implemented in or loaded into computers shown in FIG. 2, like 203 and 205, in routers at nodes like 231 and 233 of network 200, gateways connected to PSTN 285, in cellular telephone base stations 283 and 287, and in cellular telephones 281 and 289 themselves. In web television sets, and mobile web TVs, tuners 495 and 795 are included to drive display 267.1 and 267.i in the systems.

[0034] In one type of base station networking embodiment, the base stations 283 and 287 of FIG. 2 are respectively coupled directly to the packet network 200 via their own gateways 291 and 293. Base stations 283 and 287 thus communicate by VoP or VoIP over the packet network 200 and bypass PSTN 285.

[0035] Cell phones 281 and 289 also use CDP cellular digital packet data to send datagrams over packet network 100. They are further improved as disclosed herein to send VoIP or VOP datagrams at a sufficient data rate and with packet network path diversity for high QoS. The cell phone constitutes a physical layer interface (PHY) which is complemented by higher layer software as in FIGS. 5 and 6 to make it a VoP or VoIP phone.

[0036] In the cell phone, the software of FIGS. 5 and 6 is manufactured or downloaded into the unit. Then the microphone 161.1, keyboard 163.1 or .i, monitor 167.1 or .i, and speaker 162.i of FIGS. 5 and 6 are respectively replaced by FIG. 2 cell phone 281 microphone 281.4, manual input 281.1, visual interface 281.3 and speaker 281.5. In this way, an advantageous cell phone embodiment is constituted for packet network enhanced QoS VoP and VoIP and other media packet communications.

[0037] The cell phones 281 and 289 are suitably provided with positioning software such as GPS (global positioning software), or the like. The cell phones have a wearable mobile enclosure with a belt-clip 281.9 and 289.9, and their circuitry is suitably mounted in an automotive enclosure such as in the Auto shown in FIG. 2. PCS (Personal Communicator System) wristband apparatus and other highly mobile embodiments with voice-recognition control of the blocks are also contemplated.

[0038] The software process blocks of FIGS. 5 and 6 are partitioned to a microcontroller and to a DSP according to speed, power, economic and other tradeoffs as the skilled worker suitably elects. Speech codec and modem suitably run on the DSP. The TCP/UDP/IP stack runs on a DSP but suitably also is partitioned instead into the microcontroller.

[0039] In systems where a cell phone 289 communicates voice wirelessly to its base station 287, the base station recovers the voice via a wireless communications interface 331 and DSP 351 of FIG. 4. Then according to improvements contemplated here, the voice is recoded by the recoder of FIG. 5 and base station 287 uses the rest of the software blocks of FIGS. 5 and 6 to send packets onto the packet network 200 of FIG. 2. In the reverse direction, as illustrated in FIGS. 5 and 6 software is reciprocally provided.

[0040] In a further network and system infrastructure embodiment, a VoIP Solution Provider improves gateways 291 and 293 with the software of FIGS. 5 and 6 for packet network path diversity communications. Then cell phone users and cellular telephone base station operators of equipment unimproved by software of FIGS. 5 and 6 couple their equipment to improved gateways 291 and 293. The gateways 291 and 293 are also suitably provided as, or added as an add-in printed wiring board or card into, one or more private branch exchanges (PBXs). For large service volumes, as dozens, hundreds or thousands of simultaneous calls, the software of FIGS. 5 and 6 and elsewhere herein implemented in gateways 191 and 193 and such PBXs is straightforwardly made to have multichannel service, by running many voice calls with multichannel speech codecs and multichannel VoIP control for each call. Keyboard 263.i and monitor 267.i interface to the software of FIGS. 5 and 6 for occasional supervisory monitoring and control of the multichannel service.

[0041] In FIG. 5, a packet voice digital signal processor (DSP) 511 is implemented as an integrated circuit with

embedded software establishing blocks as shown. The integrated circuit is suitably a CMOS DSP such as any suitable selection from the TMS320C54x, TMS320C55x or TSM320C6x DSP families, or other such families commercially available from Texas Instruments Incorporated, Dallas, Texas USA.

[0042] For example, the TMS320C54x fixed-point, DSP family is fabricated with a combination of an advanced modified Harvard architecture which has one program memory bus and three data memory buses. This processor also provides a central arithmetic logic unit which has a high degree of parallelism and application-specific hardware logic, on-chip memory. additional on-chip peripherals. This DSP provides a specialized instruction set for operational flexibility and speed of the DSP.

[0043] Separate program and data spaces allow simultaneous access to program instructions and data. Two reads and one write operation can be performed in a single cycle. Instructions with parallel store and application-specific instructions are provided. Data can be transferred between data and program spaces. The parallelism supports a powerful set of arithmetic, logic and bit-manipulation operations that can all be performed in a single machine cycle. Control mechanisms manage interrupts, repeated operations and function calling. On-chip RAM and ROM memories are provided Peripherals on-chip include serial port and HPI host port interface.

[0044] In FIG 5. integrated circuit 511 is improved with software manufactured into the ROM, or other nonvolatile, memory for implementing some part of the process embodiments. Thus, FIG. 5 emphasizes an example of software blocks manufactured into the integrated circuit 511, the hardware described hereinabove being understood. Thus, description in software parlance follows next regarding FIG. 5 wherein for example a "unit" refers primarily to a block of software although a hardware block is another suitable alternative.

[0045] In FIG 5. voice samples are supplied from an analog to digital converter (ADC) not shown, to a PCM interface 515 and converted there to pulse code modulation. Next the PCM is fed to an Echo Canceller block 517, which feeds a Gain Control block 521. Gain control 521 supplies a Voice Activity Detector 531 which detects whether voice packets or silence packets are to be generated. The output of Voice Activity Detector 531 goes to a speech coder 541 having a Voice Coding Unit. or encoder, 551. The speech coder 541 is suitably devised or implemented by the skilled worker so as to have multiple coding rate modes as contemplated herein. For one example, G.729 and Annexes with 11.8kbps, 8kbps and 6.4kbps selectable source rates sij is suitably used. Then an Ingress/Egress Control Block 581 couples the output of encoder 551 to a Packet Encapsulation Unit 571 which thereupon outputs voice packets from the DSP. Control Block 581 also feeds control signals between itself to voice coding unit 551 to accomplish functions as described herein.

[0046] On a receive path in FIG. 5 voice packets enter packet encapsulation unit 571 where they are depacketized and passed to the Ingress/Egress Control Unit 581. Control Unit 581 has software that implements process steps for ordering processing and saving packets which would be lost by conventional techniques.

[0047] The destination is suitably improved with an integrated circuit 511' (not shown) similar to or identical to integrated circuit 511 of FIG. 5.

[0048] From Packet Playout Control Unit 581. depacketized compressed voice information being received is then supplied in a controlled manner to a speech decoder 555 portion of speech coder 541. Silence packets and voice packets. suitably dejittered and compensated by use of diversity packets as improved according to any of various process embodiments herein, then are decoded by speech decoder 555 and thus played out. The speech thus played out. passes via Gain Control 521 to PCM interface and from there to a DAC (digital to analog converter) not shown which can be provided either on-chip or off-chip as the skilled worker elects. The PCM output as converted by the DAC thus reconstitutes the voice in an advantageous manner more fully satisfactory and enjoyable to the user, by virtue of the various improvements provided and discussed herein. Further, a DTMF "touch-tone" generator 591 and Tone Detector 593 handle the dialing steps for placing a VOP/VoIP telephone call to confer a comprehensive application improved as discussed herein.

[0049] In FIG. 6. the improvements are illustratively partitioned so that the RTCP is associated with MCU 381 of FIG. 3. The ingress/egress control block 581 and other FIG. 5 blocks are suitably provided in the DSP software complement for the DSPs of FIG. 3.

[0050] In FIG. 6. MCU 381 of FIG. 3 is provided with a TCP/UDP/IP stack 611 which further has MAC/ARP, Ethernet driver and other network interface protocol blocks. Further, network management software 615 for MCU 381 has a network management agent controlling and interfacing to a first software block for embedded webserver HTTP (Hypertext Transfer Protocol) and Java applications, a second software block for SNMP protocol, Voice MIBs, and Protocol MIBs, and a third software block for TFTP software download. Still further, telephone signaling gateway software for MCU 381 has call processing software, address translation and parsing software, and H.323 protocols including H. 225 signaling, H.245 software, and RAS/RTCP software. The RTCP function in block 619 is coupled to the UDP function in TCP/UDP/IP stack 611 and also coupled to the Packet Encapsulation unit in DSP 511 of FIG. 5.

[0051] A DSP interface manager software block 621 is coupled to software blocks 611, 615, 619 and 623 and communicates with DSP 511 of FIG. 5 and the software blocks described in connection therewith.

[0052] MCU 381 runs system software 623 including RTOS (real time operating system such as Microsoft Windows CE or Symbian EPOC, as well as DSP 511 running BIOS™ RTOS from Texas Instruments Inc.) System software 623

includes WDT driver software, flash memory manager, BSP software, development and self-test (IPQST) software, and software installation code.

[0053] Multiple DSP embodiments of FIGS. 1 and 3 can use several C54x DSPs from Texas Instruments Incorporated, Dallas, Texas, such as in a line card having four (4) C54x DSPs. For example, a telephone central office can have 100,000 (one hundred thousand) lines for handling 10,000 (ten thousand) phone calls concurrently. Thus, numerous DSPs and line cards are used in a telephone central office. Also, the C6x DSP from Texas Instruments Incorporated, Dallas, Texas, provides miniaturization advantages.

[0054] FIG. 8 depicts a temporal model of arrival of packets. The packets to the right of the vertical dividing line marked "now" are the packets that have already arrived, while those to the left of "now" are yet to arrive in the near future. The queue 151 of packets, part of FIG. 9, is aligned with the arriving packets model along the "now" line, which separates the past from the future.

[0055] Concurrent with arrival, queuing, decoding of packets, and placing the frames in the egress buffers 181 there is still another process that is taking place. Samples from egress buffers 181 are being outputted, one sample at a time.

[0056] Looking at the buffer 181A, notice a pointer annotated "bf_{out}[NOW]." It is the address of the front of the data. The address $bf_{out,A}$ which earlier (FIG. 9) was associated with time "NOW" is pointing at the next sample to be retrieved and sent to a DAC. Another address named $bf_{in,A}$ points at the future first data word of the next packet's frame. Between the two addresses there is data ready to be sent out to create the voice stream. That data, marked "R_A" is the data reserve.

[0057] Channel B with the reserve of R_B being smaller than R_A is also shown at the right in FIG. 9. For the two channels the reserves are being depleted in the same rate (of 8,000 samples per second). Thus the state of B will become critical before the state of A does. Indeed, the reserves of B must be replenished before the state of A becomes critical.

[0058] The quality of voice communication can be improved if the order of processing of packets is made to depend on the needs of each channel.

[0059] Quantitatively, the reserve is the difference of two addresses:

$$R = bf_{out} - bf_{in} ;$$

[0060] This is the measure of the reserve in terms of the number of voice data words in the reserve. Now consider measures of time expressed as the number of clock cycles. Assuming that the clock is the sampling clock, the number of time units C_A in that region are the same. Thus $C_A = R_A$.

[0061] FIG. 10 illustrates the "FIFO memory" model of egress buffer reserves for several channels. The FIFO memory model assumes that all data is always shifted toward the output (on the left of the buffer). Each time a data element is withdrawn from the memory, all remaining data is shifted forward. When data is added, it is appended to the end of existing data.

[0062] In FIG. 10 the reserves of all the channels reserves can now be compared at a glance. Equally important, the "NOW" for all the channels is aligned: in fact it is seen as a single line. The reserves are consumed by "moving" toward the NOW, one sample at a time, at 8kHz rate.

[0063] FIG. 10 previews a design possibility, which takes advantage of the fact that the duration of all frames is a multiple of a 10msec period. That allows alignment of all frames' starting points at any one of the 10msec boundaries. Notice on FIG. 1.4 that "NOW" for all channels is away from the next 10msec boundary by the same fraction of the 10msec period, thus the same number of samples.

[0064] Aligning the starting points of the frames also aligns the ends of the frames. in consequence (inspect FIG. 10 again) all the reserves' ends fall on one of the 10msec boundaries. That implies that deadlines for any packet also fall on one of the 10msec boundaries. In consequence the time resolution of measurements can be lowered from a single sample (125usec) to the block period (10msec).

[0065] The input and output digitized signals are divided into frames. The size of the frame depends on the vocoder process, and can range from 10 msec to 40 msec. Current vocoder processes all have a greatest common divisor of 10 msec or 80 samples worth of data. Both the input as well as output sampled data frames are aligned along the common 10 msec boundary. 'Frame task,' or simply 'task' means the CPU activity on behalf of one frame for one channel.

Maximum delay in absence of preemption

[0066] For a given process, all the channels must be executable within the period of the process. The calculation determines that a process servicing 32 channels with 20ms frame size must have 20/32 ms process execution time per frame task to service each channel. If the encoder is 2/3 and the decoder is 1/3 of the time then the decode and

encode upper limits are determined.

[0067] In one set of embodiments, illustrated in FIG. 11an architecture saves moderately-late-arriving packets which would be effectively lost to the system if processed in order. The solution advances processing of the packets depending on their deadline. This type of embodiment is useful for instance in infrastructure systems where one or more real time decoders 1171 process many media channels serviced by many parallel buffers 1175.a, 1175.b, 1175.c,...1175.q that are receiving packets concurrently. A packet buffer 1181 receives packets for the many channels. If the packets were processed on a first-in-first-out basis from buffer 1181, holding packets C,A,B,Q,B,Q,B,A,A, various late-arriving packets C,A,B would be useless to the system because they would be processed too late. A selector 1185 operates according to a process that advances processing of a packet C, for example, where channel buffer 1175.c is almost empty, by moving packet C "to the head of the line" that is, by moving packet C from the tail end buffer 1181 to the almost-empty channel buffer 1175.c. Thus, the late packet C is made available to decoder(s) 1171 soon enough to be useful, or even just in time, advantageously improving the operations of the system.

REMOVING FIRST DATA PACKET DEPENDENCY IN A CHANNEL

[0068] Returning to FIG. 11, a possible problem in VoIP is that the decoder output begins and depends on the circumstances of the arrival of the very first packet of the compressed speech. This phenomenon is called first packet dependency herein. Depending on the accident of the arrival delay of that first packet relative to the succeeding packet (s) the process may propagate this accident in an unfortunate way and degrade the reception performance for all the rest of the packets. To solve this problem, a buffer 1171 is used to accumulate some packets. The decode system 1511 is made intelligent enough to delay the egress output of the first packet, thereby advantageously reserving a cushion of time for subsequent packets before their deadline expires. In this way, late arriving packets that arrive, for example, 40ms late still have 20ms of grace, thanks to the provision of the 60ms buffering of the very first packet.

[0069] This buffering time is chosen long enough to provide effective deadline cushioning, and short enough to not unacceptably contribute to delay in conversational speech that might be noticed by the users before they can hear reply speech.

[0070] In this way, the improved decode system, device and process ameliorates the sensitivity of VoIP/VoP/media system to accidents in arrival delay of the very first packet in a channel. By contrast a conventional anti-jitter buffer merely evens out the variations in delay between successive packets in the communications stream. Buffer 1171 is a both single-channel and multiple-channel improvement.

[0071] Alternatively, the first packet is simply placed back a number of spaces in its channel buffer of FIG. 11. This result is advantageously accomplished by suitably programming selector software or configuring buffer hardware.

[0072] In other embodiments, the packet buffer has a sophisticated selector process 1185 and distinct channel buffers 1175.a-.q are unnecessary.

[0073] The graph of FIG. 12 is illustrated as stairsteps, when time available for decode permits (right to left) at first only one full frame per channel of decode, then 2 frames of decode, then 3 frames of decode, etc., in the time available. The graph of FIG. 12 recognizes that sequencing the latest very late packet to the head of a sequencing queue becomes ever more critical the more execute time a frame needs for decoding, compared to the execute time the system can make available. The time available for decode graph of FIG. 12 is net of time needed by the system to perform ingress and overhead processing, which is estimated elsewhere herein. As discussed here, embodiments of sequencing advantageously provide the most advantage in the most demanding of short frame intervals, all channels active and lower performances (less expensive) DSPs.

[0074] In FIG. 13 for egress method, operations begin at Egress 1200 with an arriving packet reception step 1211. The system has an organization of channel records egr_chnl_rec used by a step 1213. The channel number points to a corresponding channel record. Step 1213 extracts the packet deadline for a given packet and updates the organization of channel records egr_chnl_rec. The channel record contains a deadline value, which is a number that RAS (remote access switching) design specifies a number of 10 millisecond (ms) units to which the packet is subject before it becomes useless and may be thrown away. Next, a step 1215 places a packet of data in an egress scheduling list egr_sched_list according to the packet deadline value. Later, a step 1217 updates the channel record egr_chnl _rec. This update step 1217 is performed suitably on 10 millisecond boundaries (or otherwise periodically) or alternatively performed on some regular basis whether periodic, non-periodic determinate, or non-periodic random.

[0075] Consider a voice stream of conversational voice. Interspersed with the voice are various spaces of silence. When the computer receives the packets and converts them into voice, some packets may be too late or lost and have to be replaced with silence with decay, noise, or interpolated data. In the case of silence, just before the D/A conversion, time constants of rise time and decay may be used. Thus, in process control systems, which might otherwise respond to silence very violently (e.g. pressure expected to be 25psi is found to be zero), "silence" or "zero" frames are handled in a way that provides appropriate rise and decay respective to the system application.

[0076] In the voice area, the silence frame or silence packet contains an amount of time of the silence, or can be

sent packet by packet.

[0077] When the silence packet enters the queue, a process embodiment here bypasses the decode process and go directly to the output side of the decoder. and make a period of silence. If a silence packet arrives late, it is not advanced in the queue as a voice packet would be, in the manner discussed earlier hereinabove in connection with step 1215. Therefore, the nature of a packet as being a voice packet or a silence packet suitably is introduced into the process

[0078] In FIG. 14, a Sequence Queue and Management block receives a voice packet 1409. A silence packet detector or selector 1405 routes voice packets such as 1409 to block 1431. The selector 1405 routes silence packets such as silence packet 1407 to a post-processing block 1441. An example of postprocessing generates voice data directly into the output buffer 1451 or into its local, or private, buffer 1461. Silence causes post processing 1441 to transfer silence directly to output buffer 1451. thus to fill certain spaces in the output buffer with data corresponding to silence in a manner consistent with the method used to represent silence in the system. The silence processing bypasses the queue block 1431. The postprocessing updates the channel records 1413 and increases the delay by the number of milliseconds of silence, thus acting as a source of maintenance of egr_chnl_records 1413.

[0079] When a silence packet is followed by consecutive voice packets, then according to schedule update in link list 1431 if 150 milliseconds of silence occur, the voice packets are scheduled in channel record 1413. The post processor simply updates by addition. If the frame is 4 units wide then the silence record (representing a frame 4 units wide; causes an update of an entry of 3 in channel record by adding 4 to 3 to equal 7. This then is the deadline interval for the next voice packet. Furthermore, if the silence packet is of a type that identifies plural frames of silence, by a number S in the packet, then the update is equal to the channel record plus 4S. (For example, 4S + 3 is the new updated deadline interval value in the channel record.) Of course, if another type of packet represents a different frame width F. the number 4 is replaced with that frame width. In general the process updates a value of DI by the formula

$$DI = DI + S \times F.$$

[0080] A packet arrives. Its character as silence or voice is detected in step 1405. Actual stripping of header, extracting data and deciding whether silence or voice may involve 50-100 instructions, and these are concisely represented as the silence packet selector 1405 diamond. A voice packet 1409 goes to the queue 1431, eventually gets sent to voice decoder processing 1425, goes to post processing and decoded voice gets into the buffer 1451, and postprocessing 1441 updates the channel record 1413. Postprocessing updates the deadline interval DI or deadline interval entry in channel record 1413. Deadline interval whose example is 3 in channel record 1413 is updated by postprocessing 1441 of FIG. 28.

[0081] Decoding process 1425 provides digitized decoded voice to local buffer 1461 and also signals postprocessing 1441 which thereupon or concurrently does maintenance of the deadline interval entry in the channel record 1413 by updating it in the following way. The number of frames S that the silence packet represents is multiplied by the frame length F indicated by example numeral 4 in channel record 1413 whereupon the product is added to the deadline interval (e.g., 3) currently in the deadline interval entry of channel record 1413. Also the postprocessing 1441, depending on whether it is processing silence or voice, initiates a local buffer 1461 to store digitized voice from decode process 1425. Or, if the packet is silence, postprocessing 1441 causes a silence word 1465 to be transferred to the output buffer 1451 in the one or more number frames indicated to be silence by the silence packet. If there is a voice packet, the postprocessing 1441 causes the local buffer 1461 to issue a voice datum into the output buffer 1451 to take its place in the queue of that output buffer 1451.

[0082] Next consider the processing of a succession of silence packet, voice packet, silence packet, voice packet. The process detects a silence packet, then postprocessing 1441 fills out the output buffer 1451 with zeroes (or other silence signal).

[0083] Turning to FIG. 15, a record in egress channel records 1413 has an example number "3" (with legend "deadline interval") entered therein indicating that it has 3 ten-millisecond intervals remaining before it becomes useless. This number "3" does not stay continuously to be 3, another component part of the channel record is a process wherein the deadline is updated regularly, e.g. every 10ms., decremented by one (1) every 10ms from 3 to 2 to 1 to 0. Thus, in a multichannel data structure, all the deadlines are reduced by one (1) every 10ms. When the decrement goes from one (1) to zero (0), then the packet is too late to use anymore.

[0084] Further in the data structure egr_chnl_rec is an entry egr_in_use variable. The variable entry is "in use," so the entry is one (1). The channel becomes "in use" during a period when the channel was free (entry is zero (0)) and then a next call has to be processed, and then a given channel is assigned to that call and then the "in use" number is set to one. Advantageously, when the process is scanning many (e.g., 32) channel records, the system processing uses the egr_in_use variable to avoid processing those channels that are not in use since there is no need to do any processing on them. In a machine that can process 32 channels, such as 32 telephone calls, if at a given time only

ten telephone calls need be processed, then only ten of the channel records would hav a one (1) entry in egr_in_use.

[0085] The next entry number (for example, shown as "4") is data frame size F in units of 10ms intervals, and is process dependent. This is not the packet length, but instead the length of a frame of compressed data, of which one or more frames are carried in the payload of a packet received in step 1211 in FIG. 13 and a packet 1125 in FIG. 1.

[0086] Beginning with raw data on the ingress side, a certain amount 40ms. of PCM data is compressed into a packet by an encode process. Usually, but not always, the packet changes the amount of data from 320 words to perhaps as few as 20 words (16:1 compression). If the channels are very congested, the system suitably chooses a different compression process that compresses a fewer number of words, say 160, to 20 words (8:1 compression). Pure voice 1111 of FIG. 2 in digitized form is encoded into packets that are sent into the network 1121. One encoder process produces a frame 40ms long. Another encoder process takes 160 words equivalent to 20ms which also produces about 20 words for packets. Thus a frame is that interval of uncompressed digitized speech which an encoder process takes as its basic unit of compression.

[0087] A "frame task" for the ingress process is a task to take a frame for the encode process and produce a packet. The encode process depends on the sending computer process. For a telephone call, the encode process used by computer 1101 to send to computer 1151 may be a different encode process used by computer 1151 to send to computer 1101.

[0088] On the egress side, a frame task is the inverse process of decoding a packet or frame into decompressed or decoded real time data. On the egress side, a frame task is the processing required to reproduce from a single packet of data the frame which was intended to be reconstructed.

[0089] Among about a dozen popular international standard codec processes, choose which of the processes to use for a given transmission (or switch between or recode processes in a single transmission). Suppose, among 30 active channels on an egress side, some of the channels use process A using 40ms frame size, others use a GPS process using 30ms frame size. still others use frame size 20ms-so that the system at any moment is processing a mix of processes. Note in FIGS. 14 and 15 in channel record 1413 having deadline numeral 3 for 30ms, that this counts clock time. Instead it is the numeral 4 that represents the 40ms frame size corresponding to a given 40ms process as just discussed.

[0090] A reserved space for a finer resolution is marked with a star (*) in block 1413 of FIG. 15.

[0091] A queue 1431 holds all the channel records. The system has the deadline information indicating by when each packet has to be executed. The system copies the deadline interval entry "3" in block 1413 and associates it with its corresponding packet. The queue is a link list 1431 in FIG. 16, designated egr_sched_cell comprised of linked cells. Each cell in the structure egr_sched_cell is a cell of three words that has a pointer to a packet and has a deadline for when that packet needs to be executed, and finally has a pointer to the next egress schedule cell in the queue, thus ordering the cells in order of their deadlines. The order of the cells represents the order in which the process is arranged to execute the decode process on the frame contents of the corresponding packets. Also note that the whole queue is advantageously updated every 10ms. Thus, at 10ms intervals a decrement process goes accesses every cell in the queue 1431 and decrements the deadline numbers by one (1, representing 10ms).

[0092] Every cell that has a deadline of zero or a negative number after the decrementing process, or upon arrival, is thrown away from the queue 1431. The output buffer to the decoder has the corresponding head-end entry left blank or zeroed out, whereupon the decoder performs its native response to the absent frame.

[0093] Thus, each cell in the que egr_sched_cell points to a packet that still needs to go through the decode process and provides the deadline for the packet.

[0094] In FIG. 15, note further that the structure 1413 egr_chnl_records may not have the deadlines in same order as ABC in queue 1431 egr_sched_cell of FIG. 16. When a data packet 1511 arrives in FIG. 15, the process first associates the packet with an empty cell 1541 (called N). The software process first assigns in cell 1541 a pointer 1545 to the data packet 1511 and thereafter the data packet 1511 need not be moved, and only the egress schedule list (queue 1431) cells of FIG. 16 are relinked as the process proceeds.

[0095] In FIGS. 15 and 17, note that the letter N does not necessarily indicate that the cell 1541 will become the last cell in the queue. Instead the cell 1541 is inserted in the following manner into the queue 1431 in a place ordinally established by its deadline value "3." Once the pointer 1545 has been established, the process reads, recovers or calculates the deadline value 3 from the packet 1511 and plugs the deadline value "3" into the middle place in the new cell 1541 as illustrated in FIG. 15. Next, the process goes to the egress schedule list queue 1431 egr_sched_cell of FIG. 16 to establish such a place (see FIG. 17) for the cell 1541 in the queue 1431 (and thus indirectly for the packet as well) that the deadlines above will be shorter, and the deadlines below will be longer, thereby advantageously establishing, maintaining and updating list 1431 as an ordered list queue.

[0096] In FIG. 16 queue 1431 represents the cell scheduling list at a time befor update with the latest cell as in FIG 17. The list 1751 is the egress scheduling list 1431 after having been updated with the Nth record 1541, except that cell 1541 has had its last (3rd) word arranged to point to cell C in old list 1431, and record B of cell list 1431 has had its 3rd word updated to point to the cell N just added. In the software the 3rd word is called NEXT. The deadline of cell

B is less than the deadline of the new cell N, so the new packet 1511 can wait longer than B, and when the deadline for the new cell N is hypothetically less than or equal to that of cell C then the new cell N is sandwiched in between cell B and cell C in the queue.

[0097] The process performs a linked sort according to any of several methods. In a one method, cells and packets are not moved in memory or other storage. Instead, the method simply updates the pointers to keep the list ordered as new packets arrive and corresponding new cells get added to the list. Further the method updates the NEXT pointers to drop old cells as packets get decoded or go past their deadline. An alternative method physically relocates data in storage under software control to keep it in a particular order. In a further alternative method, hardware or firmware accomplishes the reordering and maintenance of the queue. Lookup table maintenance routines in RAM on a DSP, various kinds of table sorts and other methods are also contemplated.

[0098] Basically, in various embodiments as the new packets come in they are entered into a deadline-order list, or cells corresponding to them are entered into a deadline-ordered queue, or they are otherwise reordered, maintained and processed, in a manner responsive to the order of their deadline information or otherwise as a function of their deadlines.

[0099] In FIG. 16, maintenance of the scheduling list 1431 involves two distinct processes. A first process decrements the time-to deadline in the queue by one, every time 10ms passes. This first process is initiated by a 10ms interrupt, and the first process goes through the queue following the NEXT pointers and decrementing every cell deadline entry by one. This is suitably performed either inside an interrupt routine or just following return from interrupt. The last pointer in the queue is NULL or zero.

[0100] In an alternative form of the queuing process for handling the last cell in the queue, a header has two pointers. One pointer points to the beginning cell of the queue, and the second pointer points to the last cell of the queue. The process traverses the queue following the NEXT pointers and also comparing each successive NEXT pointer with the second pointer in the header. When the NEXT pointer equals the second pointer in the header, then the last cell is processed, and the update process terminates.

[0101] As stated two paragraphs above, updating the queue involves two distinct processes. The second of the processes updates the queue by inclusion of a newly-arrived packet 1511 from which a deadline was extracted for block 1413 and puts the deadline into new cell 1541. Thus, the second process includes the new cell into the scheduling queue 1431. The second process accomplished this by going through the queue comparing the deadline of new cell 1541 with the deadline of each cell A, B, C already on the list. The process suitably assumes that the list is already sorted in deadline order, and then writes into the NEXT field of the new cell 1541 only when the deadlines higher in the list are smaller than the deadline "3" in new cell 1541 and the deadlines lower in the list are greater than or equal to the deadline "3" in new cell 1541. In this way the new cell 1541 is "inserted" into the list 1431.

[0102] Note further that some deadlines will be equal for distinct packets, so that the order of these distinct packets or cells in a larger queue may not matter, and they are suitably listed together in the larger queue. Alternatively, and advantageously, layers of priority are introduced in addition to the deadline priority layer just described. For example, frame size is a next deadline priority layer. Long frame size is advantageously given a higher priority than shorter frame size, because losing a 40ms frame is more serious than losing a 10ms frame.

[0103] The identity of the caller (by company, by organizational position, telephone operator, by name, or by service feature such as call waiting) is suitably introduced as a third layer of priority, thereby to put the cell with deadline "3" in a position in the queue relative to other cells having the same deadline "3" in accordance with this third layer of priority. Thus, yet another set of permutations of the process can be used to the put the cells in a desired order.

[0104] The foregoing considerations suggest a rule, or further dimensions of ordering the queue 1431, in the part of the process that orders the queue in a second key of ordering.

[0105] All of the packets in the queue egr_sched_cell are assumed to be in the same channel.

[0106] Sometimes, the packets from the same channel arrive in the wrong order. This is handled by reordering according to UDP (Universal Datagram Protocol) sequence number in the header of the arriving packet of FIG. 7. If packets are coming out of order, then the process determines the (large) number of milliseconds for a very-early-arriving packet and creates a new cell 1541 that will be rather low in the queue. When a subsequent out-of-order packet comes later for this same channel, another new cell is created with a relatively short deadline pertaining to the out-of-order packet to which the cell corresponds, and that new cell is entered into the queue higher in the queue than the cell for the earlier-received very-early-arriving packet. Thus, the sequence number or time stamp of the arriving packet is used in computing the deadline.

DEADLINE INTERVAL DI

[0107] Where a voice decoder has a FIFO ahead, determine the hard real time to deadline, also called "deadline interval", DI herein, which includes the processing execution time E required. For an arriving packet N the deadline interval DI_N is the difference between arrival time A and the deadline instant D_N by which the particular channel must

receive new data or suffer a frame of silence. If a packet has plural frames with different deadlines, the deadline interval DI is computed for each frame from time "NOW" to its respective deadline. As time elapses DI decreases, as the "NOW" approaches deadline D.

[0108] The formula for computing deadline interval DI in the scheduler often need not be revised if there is an egress buffer (called Egress Buffer in FIG. 1 and Egress Channel Buffer in FIG. 9) following the voice decoder. However, some embodiments do make the sample FIFO run asynchronously to the decoder, or insertion of very late out of order packets into the sample FIFO then the deadline interval DI should be computed relative to deadline instant for sample FIFO output and not relative to the deadline instant for frame output from the egress packet queue that feeds the decoder in FIGS. 1 and 9 (and called channel buffer in FIG. 11).

[0109] A recursive first procedure for computing the cell deadline entry D for new cell j 1541 is given as:

$$DI_j = A - [\text{ (Deadline D entered in queue for packet with highest}$$

sequence number Si for given channel C) + (Frame size F for

process being used in egress channel C) x (Sj - Si - 1)]

where Sj is the sequence number of the out-of-order packet.

[0110] $DI_j = A_j - (T_0 + (i - 1)F) - (S_j - S_i - 1)F$ as further described below.

[0111] This first procedure depends on calculating deadline D for an original first packet when no cell has yet been established in the queue. Thus, the deadline D for that original first arriving packet is computed as the time when the decoder is first ready to accept that first packet minus the time of arrival A of that first packet. The deadline for each subsequently arriving packet is then computed from the formula. Note further that if the first packet was out of order, then the lower sequence number of a later-arriving in-order packet will produce a negative number for (Sj-Si-1) and thus may cause one or a few packets to be (acceptably) lost by virtue of passing their deadlines at the beginning of the transmission. This first procedure, being recursive, works well when the process of maintaining the queue has high reliability and low error rate. Error checking and redundant storage in the queue keeps errors low.

[0112] A second procedure for computing the deadline provides a storage location for the time To when the decoder was first ready to accept the first packet, and then continually increments that storage location with the frame size F in 10ms units. For the ith packet, the decoder will be ready to accept that packet i at a time $T = To + (i-1)F$. The deadline interval DI_i is the latest packet time of arrival A_i less the decoder-ready time D_i. Be careful not to confuse deadline interval DI_i with deadline instant D_i. Thus, $DI_i = A_i - D_i$. In this second procedure, the formula for the cell deadline interval entry is

$$DI_i = A_i - (T_0 + (i - 1)F.$$

[0113] This second procedure works well when there is substantial coherence or synchrony between the clock that produces the incrementing on indices i and j, and the clock that produces the sequence numbers S.

[0114] A third procedure uses both the first and second procedures, with error checking, for high even higher reliability when desired.

[0115] Further the process drops packets that remain beyond their deadline as decremented. There is no point in processing such packets, and they may be discarded. Assume the process waits for the out-of-order packet. There is a moment in time in which the packet can be lost. If the packet is lost, the decrementing process in the queue determines when it is lost. When the time has passed for the lost packet to arrive, and it is still not there, then the time has come for the decoder to put out blanks or noise into the audio stream, or otherwise do what the decoder is established to do reconstruct or otherwise handle lost packets.

[0116] In a first approach to handling out-of-order packets, the process detects sequence number (tag, time stamp, etc.) and channel number in the header of arriving packet 1511. The process goes through the process as described above; however, the out-of-order packet to be decoded has a deadline in the channel record 1413 equal to the deadline interval ("3") plus the frame size number ("4") multiplied by the number of missing sequence numbers. So, if the out-of-order packet has one currently-missing packet in between the out-of-order packet and the most recently previously received packet in the same channel, then the deadline number is entered to the queue 1431 as a cell with deadline revised to be 3+1x4 =7. Thus, in a system with a sorted queue and known frame length for a given process, this process confers an elegant and advantageous solution.

[0117] In a second approach, a separate storage area holds the out-of-order packet (e.g., in a stack) until the next in-order sequence numbered packet does in fact arrive. The in-order packet is issued a new cell 1541 and new cell

1541 is entered into the queue 1431. Subsequently, the out-of-order packet in the separate storage area is then issued its own new cell, and that new cell is additionally entered into the queue.

[0118] One form of the process insures that additional packet will not be placed, the data coming from the coding on the extra packet, will be placed into the data stream remembering the 40ms break. When a packet is out of order, the process not only adds 4 units in block 1413 but also schedules in the cell in the queue 1431 so that when the data is put into the decoder buffer, e.g., 1175.c, the data is placed farther back at position 1291, leaving one or more positions 1293 empty. The number of empty positions 1293 are equal in number to the quantity (Sj - Si-1).

[0119] In FIG. 1, an ingress buffer 121 is storage for at least two frames in order to have a complete frame of digitized voice 1111 to process by encoder 131, say 40ms of data on the ingress side. The data is arriving and eventually the ingress side collects 40ms of data. Then the system schedules the ingress process for encoding that 40ms into a packet to be sent out. In the meantime, the system has space in ingress buffer 121 for a second frame because the data is coming in continually. Also, more than two frames are suitably used, but at least one buffer space has a frame being processed by the encoder 131, while one or more buffer spaces in ingress buffer 121 are provided for further data 1111 coming in.

[0120] For the ingress output buffers called ingress packet queue 141, storage for at least two frames is provided. When there is no jitter, two frames would be sufficient. When jitter and even out-of-order packets are present, buffer space for more than two frames is advantageous.

[0121] In FIG. 18, in a more complex sorting process, the deadline queue 1431 is augmented with an additional key that takes into account the amount of time that the decoder takes to work.

[0122] Once the list is sorted in order of time until deadline, then the different frames are destined for different decoders one 40ms, another 30ms, 20ms. The FIG. 12 process decides which frame to issue next to the decoder in a single DSP embodiment wherein only one decoder is invoked and executing at a time.

[0123] Scenario #1: At the bottom of the link list queue 1431, suppose a cell has a deadline entry of 10ms intended for a 20ms decoder and higher in the queue is a cell with a deadline entry of 10ms is destined for a 30ms decoder.

[0124] Scenario #2: At the bottom of the link list queue 1431, a cell has a deadline entry of 30ms intended for a 10ms decoder and higher in the queue is a cell with a deadline entry of 10ms is destined for a 30ms decoder.

[0125] Scenario #3: At the bottom of the link list queue 1431, a cell has a deadline entry of 30ms intended for a 10ms decoder and higher in the queue is a cell with a deadline entry of 10ms is destined for a 40ms decoder.

[0126] As the process sorts, the packet 1311 the last arriving packet is the last in a given group. If the new cell 1541 corresponding to packet 1511 has deadline 3, new cell 1541 is situated or defaults in queue 1431 at the very bottom of the group of cells having deadline 3 in queue 1431 but just above and before the first cell, if any, having deadline 4 in queue 1431. The process knows which decoder to call because each packet 1653 in FIG. 16 to which the first cell byte pkt_addr points. contains channel information, and block 1413 contains the process information whether obtained directly or indirectly from the data structures.

[0127] Return to the link list 1431. The link list update process orders the packets in order of deadline interval DI. Logically those packets should be executed first which have deadline intervals in the last 10 milliseconds before deadline. So the first to executed are the ones (all records which Di=1), the next are the twos (all records DI = 2), the next are the threes (DI=3), etc.

[0128] In FIG. 18, a scheduling embodiment has actual separately maintained queues for 10, 20, 30, 40 and more, milliseconds, respectively. That way, searching and sorting the queues is faster than for one large queue made of all the separate queues combined, but the ordering principle is suitably made the same either way. Primary sort key is the deadline interval DI. Secondary sort keys contemplated herein are 1) identity of sender, 2) frame length for given process (wherein processing 40 ms frame is more important than 10 ms frame). In the 10ms group, entries in number for up to the number of channels. These items are sent to decoder in order of priority.

[0129] If the number of microseconds (e.g. 423) needed to do an egress task were underestimated, then all that would happen is that the system would lose a few packets down deep in the 10 ms queue because at that point the maintenance would decrement all the time intervals and throw away everything that had not been processed in the 10ms queue. If the decoder were slower in its operation than what it was being asked to do to process the full number N of channels, then an N-channel system would not be there in the first place.

[0130] Queues can be provided for every channel, load up the queue with two or three packets before the process commences in FIGS. 1, 9, 11 and 14. Given 32 channel queues software provides a link list for each channel respectively. Another way, puts all the channels into one queue link list. In FIG. 18, a third way provides separate link lists for 0-10ms, 10-20ms, 20-30ms, 30+ms.

SYNCHRONOUS AND ASYNCHRONOUS DECODERS

[0131] In a multiple DSP mbodiment, or multiple data path embodiment, then mor than one decoder can be simultaneously executing. In one subtype of the multiple data path embodiments, the channel decoders all begin on a

10ms boundary. This is called synchronous.

[0132] Yet a further embodiment has asynchronous channels wherein the channel decode does not start on a 10ms boundary even though the GCD of the frames is 10ms. Running the channels asynchronously provides advantages of software management of the decode processes. Here, the deadline entries in the cells are suitably provided in more granular fashion (e.g. numbers pertaining to 5ms, 1ms, or a fraction of 1ms in various versions), and still represent the time remaining until the frame is needed by its respective decode process. What is different is that the respective decode processes do not all begin on the same GCD time boundary. Thus, the channels could be synchronized on 1ms, or say a 125usec boundary in which the channel decode starts, for which deadline numbers are provided with higher resolution. Thus, the GCD 10ms approach is not required, but does provide one group of elegant embodiments.

[0133] Foreground processing does not disturb the background processing thus advantageously a fine degree of isolation. Transfer from incoming packet all the way to the queue 1431 either operates on a high priority level.

[0134] Foreground processes are higher priority processes such as those that are initiated by an interrupt. For example, host 101 writes a packet to memory by stealing computer cycles and sending an interrupt to the DSP C6201. (Alternatively, the DSP polls for a host write to determine when the host write is occurring.) Everything stops for the foreground process. At the same time, the main process calls the interrupt routine, stops the main process while the interrupt routine runs, and then processing returns from interrupt to resume the main process. The interrupt routine is written to take a relatively short time to execute, and this is called foreground processing when it has a higher priority for processor resources as opposed to the main process in the background.

[0135] Host updates egr_chnl_rec records including channel record 1413 by one process. Another process adds new cells 1541 to queue 1431. A further process decrements the queue cell deadlines. A still further process throws away cells and packets that have gone past their deadline. Yet another process issues packets based on the highest queue cells to the decoder process, which is still another process. Numerous choices are available to the real-time processing engineer skilled worker- for example, what interrupts shall be host interrupts, which shall be clock interrupts, what hierarchy of interrupt priorities shall be, what processes shall run in the foreground and background, and what routines shall take how long relative to one another. Many alternative process, device and system embodiments result from these choices, and it is unnecessary to belabor the explanation of these alternative embodiments further.

[0136] In FIGS. 14-17 when the packet arrives, the whole process of looking up channel data and updating queue 1431 could all be done at the same time. However, if the interrupt breaks the normal sequence, just as the top cell of the queue 1431 is being taken off to feed a decode process, then a frame might be lost. To avoid this problem, the process of taking off the top cell of the queue 1431 also includes a step of disabling or masking interrupts. Also, the interrupt routines are kept advantageously short. In this way, conflicts between different processes are made negligible. Also, when the link list is being updated and pointer values are being changed, the link list might be broken if an interrupt routine intervenes just then. Assign the pointer so the list is not broken, and then move the pointer down, and this part of the process is done, and the element is free-in this way list integrity is preserved. Notice this is a good reason for making the process short, so that a cell can be detached with just two or three program statements. If there were too many statements. it might otherwise be necessary to disable the interrupts which desirably is avoided for the most part.

[0137] Maintenance of queue 1431 by decrementing deadlines every 10ms is suitably triggered by a 10ms clock interrupt from the DMA (direct memory access) hardware. The DMA is programmed for the ingress side to continually put data from a T1 line (1.544 Mbps telephony communications line) into the buffer or buffers. The egress side DMA takes the data from the buffer on a continuous basis and puts the data on a T1 line going out of the system. Every 80 samples is just interrupt. Notice the DMA is running by cycle stealing, and the interrupt can take place while the DMA is still running.

[0138] In FIG. 19 cycle stealing is a process which allows the system to run two blocks at the same time, such as the microprocessor MPU and DMA working off the same memory MEM which MPU and DMA share on the same bus. The processor does arithmetic while DMA uses the bus to talk to MEM, and then MPU accesses MEM while DMA is inactive. The bus has address lines, data lines, and control lines. Without DMA, MPU has every bus cycle available to it. When DMA needs a bus cycle to read or to write to MEM, then DMA pulls one of the control lines to deny the bus one cycle to the processor. Cycle stealing is this process, and is performed by hardware.

[0139] Keyboard entries by user are important but few and far between compared to the data processing tasks of the MPU. Keyboard triggers I/O in the foreground, while background heavy duty calculations proceed, which can be interrupted without any problem.

[0140] Independently, a process called Updating queue 1431 adds each new cell 1541. The updating process is initiated by host interrupt responsive to reception of a new packet.

[0141] FIGS. 19 and 20 show the interrupt signals, their priority and what signals go to what interrupts in FIG. 15-17. The processor is suitably interrupted by a 10ms interrupt and an arriving-packet interrupt pck_in_intrr.

[0142] In FIG. 19, a sampling clock to an I/O stream DMA takes the data from 24 voice channels into a twisted pair T1 line of USA. Europe has a 32-channel E1 standard. The system takes the I/O data and puts it into memory. DMA clock produces the 10ms (80 samples, 8KHz standard telephony sampling rate) sampling clock.

[0143] In FIG. 19, the second priority channel management interrupt occurs when there is a boundary between the DSP processor and the host. The host is the control of the packets with headers across the boundary between host and DSP. In the case of this example, the boundary is the PCI bus, to which the DSP TMS320C 6201 is connected. The packet header has channel number, and an optional UDI tag or sequence number.

[0144] In FIG. 19 egress packet interrupt completes the interrupt set of the example. The egress packet interrupt creates a secondary queue 1561 using cell 1541. All the FIG. 15-17 actions take place, the deadline is looked up in record 1413, and the cell 1541 is put into the scheduling list queue 1431 in the order specified by the a particular ordering process embodiment such as deadline ordering with or without further priority layers, as described hereinabove.

[0145] If during a certain period of time, the packet interrupt occurs, then a secondary queue 1561 is created. Advantageously, this process sets up a secondary queue which exists only during the time when the frame task is processed. The secondary queue 1561 in FIG. 15 has a cell 1565 pointing to a data packet 1567. The secondary queue 1561 is identical in structure to queue 1431 by virtue of being a queue, but notice that during this time no deadline lookup on block 1413 has been made, and no scheduling has occurred. Thus, secondary queue 1561 process embodiment is not identical to data structure 1751 wherein in 1751 a new cell N has been added in deadline order by updating.

[0146] While some embodiment processes do not use the secondary queue, the secondary queue approach does provide an elegant solution with its own advantages, the reasoning for which is described here. When the packet interrupt happens, the process desirably receives the packet, and could be designed to directly embark on the update process that produces the updated queue 1251. One update process might disable interrupts, then update queue 1631, then re-enable interrupts. This approach is feasible, but has an elegant alternative.

[0147] In such alternative, the secondary queue process does very little at first by just linking, concatenating or hanging the new cell 1565 onto the bottom of a copy of queue 1431, to produce secondary queue 1561, so as not to disturb queue 1431 itself. Then the process runs an interruptible process on the list, even though the linkages are not yet in order, and operate on the list 1561 to put it in the desired order, whereupon the list 1561 is then substituted in one brief step for queue 1431 to establish queue 1751 by substitution for queue 1431 outright. In this way queue 1431 is updated to produce queue 1751 in a way that is not disrupted by interrupts and does not need to disable interrupts at any time.

[0148] Turning to the 10ms DMA interrupt and output to the decoder, this 10ms interrupt sets the rhythm of the process. device and system and is higher priority than the egress packet interrupt. The process resets the boundary, schedules the ingress (which is outside this part of the discussion), and updates all of the egress deadlines. Now, the process checks whether a new egress packet flag is set. The interrupt signal sets the flag and leads to execution of the ISR for new packet to create new cell 1541.

[0149] In FIG. 15, re-linking egress packets is the name of the function moving the new packets in from one boundary to another, and updating queue 1431 is done at a high priority. This involves a secondary queue 1561 to which only the ISR has access. Queue 1431 is suitably removed from the domain of the ISR (Interrupt Service Routine) and is never touched except when interrupts are disabled, and then new packets can be added using secondary queue 1561 without affecting the rest of the system.

PREEMPTION EMBODIMENTS

[0150] It is important to determine which packets to process first: egress packet or ingress packet. When the packets arrive in sequence, preemptive processing suitably gives priority to the egress channel. In a VoP/VoIP phone, suddenly a late packet arrives to the egress process. As improved herein, the process knows the packet is late because it has a deadline interval measuring process. Further, the improved process preempts (interrupts) the system to allow the egress packet to go to decode in time.

[0151] In FIG. 20 various preemptive embodiments, #1, #2, #3, #4 and #5, the egress packets have priority and they preempt the processor, and if egress packets accumulate they simply wait in turn until they are processed. Note, however, in least complex embodiment #1 in which the execution is simply in order of arrival, unneeded too-late packets (far right column) may be processed at the expense of not-too-late packets that are waiting but are merely Early, Late or Very Late.

[0152] Accordingly, FIG. 20 depicts another advantageous preemption embodiment #2. Each individual egress packet is analyzed for degree of lateness, and possibility of recovery by preempting system operation in favor of egress processing just in time before the deadline expires (thus ignoring Too-Late packets in FIG. 20). BIOS™ real time operating system (RTOS) for DSP provides a small real-time kernel which is a very slimmed down form of a real time operating system (not to be confused with a Bios in a PC ROM). BIOS™ RTOS has an I/O capability and a preemption of a very specific type. Once a task is assigned a priority level, the priority is essentially fixed unless the system is redesigned using a configuration tool. Each task is assigned its own single stack holding area so that storage is hier-

archical.

[0153] In this embodiment #2 BIOS™ RTOS is combined as a wrapper with the ingress/egress processing. Using BIOS™ RTOS (1K of code) to organize the scheduling, gaining information about the packet, and preempting the system for egress saves about-to-be-lost packets. The arriving packet runs a routine under BIOS™ to determine whether preemption is necessary (packet is Early, Late or Very Late, but not Too Late in FIG. 20), and then preempts to save packets

[0154] Another process embodiment features fine resolution when all the channels are out of sync. The fine resolution orders the link lists according to exact amount of time to deadline if all the decoder processes are running out of sync with one another. The process records and updates a set of different deadline times for each decoder process running on the system to compute the deadline interval when the decoder processes run out of sync with each other (i.e., 10 millisecond intervals in each decoder are staggered and thus end at different instants). In this way, the sequence queue 1431 still advantageously keeps track of DI of packets from channel to channel since DI is properly calculated for each based on the deadline instant respective to that channel decode process. This fine resolution process is suitably combined with any of the processes of FIG. 20.

[0155] Suppose all frame tasks are 20 milliseconds, with 500 microseconds processing time in the decoder and 32 channels. The time needed to process the channels contends or interferes with the need to process late-arriving new packets.

[0156] In FIG. 20 preemptive embodiment #2 advantageously utilizes relatively slow, less expensive DSP processors for processing many channels. This embodiment features limited use of preemption for very late packets in a slow system, say using one or more C54x DSPs in a line card for four channels. For four channels, while the basic time per process is the same the amount of time allotted for the process for slower processors would be not 500 microseconds but three (3) to five (5) milliseconds (ten times as much). In that case, the preemption becomes more important. The egress packets have priority and they are still ordered according to the time interval remaining to deadline, but simply all of them have priority over the ingress packet. Thus egress is not random but ordered according the need for the egress packets to be executed or processed. This embodiment is particularly advantageous for smaller inexpensive systems in which case the preemptive BIOS™ RTOS is suitably used.

[0157] In embodiment #3 of FIG. 20 organize processing with egress packets still organized according to priority and still deal with emergent situations but preempt any ingress packet when any egress packet is available to be processed in the last 10 millisecond period (Very Late category only). The Late packets (20 milliseconds) and Early packets (30 milliseconds or more) are not used to preempt, and non-preemptive processing is used to handle these Late and Early packets.

[0158] FIG. 20 embodiment #4 advantageously uses preemption to process Very Late packets at a higher interrupt priority than Late and Early packets (priority 2).

[0159] FIG. 20 embodiment #5 is described in detail in connection with FIGS. 4 and 31.

PREEMPTIVELY HANDLING "VERY LATE" PACKETS

[0160] Before reading FIG. 21-25 consider the processing limit (vertical dotted line in FIG. 12). This limit becomes crucial to saving a Very Late packet N from being lost. Even 500 microseconds of decode execution time E_M (for an Mth packet in decode) is important here. Suppose a Very Late (FIG. 20) egress frame N arrives with DI_N only 600 microseconds of life left and soon risks being thrown away regardless of the sequence queue processing. Further assume that decode processing execute time E_N for this packet frame requires 200 microseconds, leaving less time (600-200=400, which is less than E_M 500) than the total execute time E_M of 500 microseconds for a decode in progress for another channel, say. This amounts to a preemption decision criterion $DI_N-E_N>E_M$ (or $E_M+E_N<DI_N$), which is still further refined below. Preemption of the egress processing in the other channel can save the day by task preemption (FIG. 20 embodiments #3, #4 and #5) when the actual time T needed to complete the egress decode processing in progress is sufficiently advanced as described next.

[0161] Interrupt computation time is suitably also taken into account in the calculations in some embodiments, so that if processing the interrupt will occupy enough time that there is no point in handling a Very Late packet, the interrupt is not taken or is aborted.

[0162] Note that egress processing in the other channel may be at an intermediate point. The full 500 microseconds may not be needed to complete the egress processing of that channel because some time T=400 has elapsed, and decode execution is some fraction of the way (say 80%) to its own completion of E_M=500 microseconds of processing. Thus, suppose only 100 microseconds are needed to complete the decode in progress. Then the very late packet in the waiting channel has a chance to proceed, succeed in decode, and be saved from loss, since 100 + 200 = 300 which is less than the 600 microsecond deadline interval of the example. In other terms, $(E_M-T)+E_N<DI_M$. Implicit in these examples is the criterion in FIG. 20 that packet N is not Too Late, i.e., $DI_N-E_N>0$. The deadline for the new packet N is sooner than the deadline for the packet M in decode. Otherwise the packet M decode would not be preempted in

this embodiment. Since the longer time DI_M to deadline is what is relevant for packet M, the preemption decision is written $D_M > E_M - T + E_N$. Thus, the processing in various ways can be designed (FIGS. 22, 23 and 24) to signal how many microseconds remain in the channel decode processing currently underway and preemption is based on the decision criterion ($DI_N - E_N > 0$ AND $DI_M - E_M + T - E_N > 0$).

[0163] These considerations lead to Embodiments #3, #4 and #5 of FIG. 20 that do special handling of Very Late packets (FIG. 20, third column). Here processing provides information back to the scheduler telling where the decode process currently is in its operations, thereby to make a decision to preempt or not to preempt or just to throw away the packet. Embodiments for keeping time on this very fine time scale are disclosed next in FIGS. 21, 22, 23, 24 and 25.

[0164] FIG. 21 compares and contrasts channel decode processes of a Same-Deadline type and a Staggered Deadline type. At the top of FIG. 21, egress and ingress for all 32 channels are handled in a 10 (ten) millisecond window 2105 that extends from left to right across the page. All the channel decodes in the Same-Deadline channel decode process type need to be finished by an instant 2111 of the 10 millisecond Deadline. The execution order of various channel decode executions e1, e2, ... e31 for the 32 channels is relatively unimportant. If a packet for a given channel, e.g., channel 7, has not arrived, then the channel 8 decode is performed. When the channel 7 packet does arrive, then it gets decoded, occupying execution interval e7. The ingress execution (Ingress Overhead interval) is suitably performed for the channels at beginning, at the end, or sometime in between, such as when not all channels have been decoded, and the system is waiting for more packets to arrive which have deadline 2111. Also, packets with a later deadline than deadline 2111 are suitably decoded, when there is time available for them. Also, such embodiment responds to very late arriving packets that do have deadline 2111 by preempting packets and packet decodes that have a later deadline than deadline 2111.

[0165] Compare now Staggered Deadline processes having their deadlines Di spaced along a 10 millisecond window 2121 having a clock boundary 2125. No longer does the clock boundary 2125 represent a single same deadline for all the 32 channels. Now, decode execution order for the various channels becomes important even for packets whose decode deadlines lie in the window 2121. Very-late execution order is important, and very late packets in some embodiments are made preempt other very late packets, as well as late and early packets, in order to reduce incidence of avoidable packet loss.

[0166] In FIG. 21 beneath window 2121, one example of the advantageously intelligent decode execution of packets in a Staggered Deadline process embodiment begins by recognizing the order of the deadlines Di in window 2121. For example, suppose the deadlines come in the order D4, D9, D3, D5, D1, D2 for illustratively six of the 32 channels. Packet 4 (meaning the channel 4 packet having deadline D4) is present, and the embodiment launches decode execution of packet 4, which occupies execution time interval E4 which completes not only ahead of the deadline D4, but coincidentally also prior to the instant D4-E4 when execution of packet 4 would have had to commence to save the packet 4 from loss.

[0167] Decode execution of packet 4 being completed, operations proceed to detect the presence of packet 9, which turns out to be present. The embodiment launches decode execution of packet 9, which occupies execution time interval E9 and completes ahead of the deadline D9, and interestingly after the instant D9-E9, the latest time packet 9 would have had to commence decode.

[0168] Decode execution of packet 9 being completed, operations proceed to a detection step to test for the presence of packet 3. However, the detection step determines that packet 3 has not arrived. Thereupon, operations test for presence of packet 5 which has the next most urgent deadline D5 after deadline D3. The detection step determines that packet 5 is present, and decode execution of packet 5 is launched (LAUNCH 5) quite significantly before its deadline D5.

[0169] Suddenly, packet 3 arrives very late. The embodiment responds to packet 3 by calculating that sufficient time exists to save packet 3, and therefore proceeds to preempt the now-underway decode execution of packet 5. Operations in FIG. 21 moving upward along the arrow PACKET 3 SAVED to LAUNCH 3 representing the operation of launching decode execution of packet 3. (Note that packet 5 decode is partially complete and these results are saved temporarily.) Decode execution of packet 3 occupies execution time interval E3 which successfully completes before deadline D3 and incidentally after the latest instant D3-E3 when packet 3 decode would have had to commence.

[0170] Decode execution of packet 3 being completed, operations do not need to proceed to the detection step to test for the presence of packet 5. Instead, operations RETURN to the point where decode execution of packet 5 was interrupted earlier, whereupon decode execution of packet expends the balance of execution interval E5, and completes decode not only ahead of deadline D5 but even ahead of instant D5-E5.

[0171] Decode execution of packet 5 being completed, operations proceed to the detection step to test for the presence of packet 1 which has the next-most-urgent deadline D1. However, the detection step determines that packet 1 has not arrived. Thereupon, operations test for presence of packet 2 which has the subsequently-next most urgent deadline D2 after deadline D1. The detection step determines that packet 2 is present, and decode execution of packet 2 is launched (LAUNCH 2) well before its deadlin D5.

[0172] Suddenly, packet 1 arrives very late, see SUDDEN ARRIVAL OF PACKET 1. The embodiment responds to

packet 1 by calculating that sufficient time exists to save packet 1, and therefore proceeds to preempt the now-underway decode execution of packet 2. Operations in FIG. 21 move upward along a vertical arrow to launch decode execution of packet 1 fortuitously just at the last instant D1-E1 available for packet 1 decode to commence. (Note that packet 2 decode is partially complete and these results are saved temporarily.) Decode execution of packet 1 occupies execution time interval E1 which successfully completes precisely at deadline D1 (PACKET 1 SAVED).

[0173] Operations now RETURN to the interrupted decode execution of packet 2 at the point RESUME 2, whereupon the balance of execution time interval E2 is expended, and whereupon decode of packet 2 successfully completes illustratively, fortuitously and precisely at deadline D2.

[0174] FIGS. 22, 23, and 24 illustrate three different embodiments for accomplishing interrupt (preemption) processes for saving packets as illustrated in FIG. 21. Where similarity or identity of function of blocks in FIGS. 22, 23, and 24 is disclosed, the last two digits of the numerals are made the same from Figure to Figure. To better comprehend each of the FIGS. 22, 23, 24, the reader is advised to study the descriptions of all of them, and compare and contrast with the last two digits of numerals in mind.

[0175] The skilled worker prepares any of a variety of embodiments to accomplish the functionality of FIG. 21 and the like, allocating process steps to foreground and background processing, or to main and preemption or interrupt processing in a manner effective to accomplish these functions and suggested in FIGS. 22, 23 and 24. Testing and debugging are, of course, inherent in the development process to effectuate the teachings herein.

[0176] In FIG. 22, a first process embodiment encounters arrival of a packet 2201 having Frame in channel N of decodable real-time information, and in a step 2205 receives, depacketizes, and enters the frame in channel N into an Egress Packet Queue of FIG. 9. Concurrently, a decoder process 2211 for a frame in different channel M is in progress to produce a Decoded Channel Output. The process provides a timer 2215 of elapsed time of the decoder process 2211. wherein the decoder process sets up the timer 2215, and sets the timer 2215 running.

[0177] Meanwhile, due to arrival of the new packet (see e.g. SUDDEN ARRIVAL OF PACKET 3 or 1 in FIG. 21), high priority interrupt process steps begin at BEGIN 2221. Next, a step 2225 computes Deadline Intervals DI for the frames in channels M and N. The interrupt task accesses the timer by input of Elapsed Time T in a step 2229.

[0178] A decision step 2231 determines whether there is enough time to execute both packets before their respective deadlines. (Compare with the part of FIG. 21 involving LAUNCH 3 and LAUNCH 5 wherein packet 3 is saved.) For channel N, e.g. channel 3, Deadline Interval DI less the decoder decode execution time EN required to decode a frame in that channel N is tested for greater than zero as a First Condition. For channel M, e.g. channel 5, Deadline Interval DI less the balance of decoder decode execution time required to finish decoding a frame in that channel M is computed less the time EN to be spent decoding channel N during the interrupt: as a Second Condition. The just-mentioned balance of decoder decode execution time required to finish decoding a frame in that channel M is EM minus Elapsed Time T. Thus, in the algebra the second condition becomes DIM - EM + T - EN > 0.

[0179] In decision step 2231, if BOTH the First Condition and the Second Condition are true (YES), then operations proceed to a step 2235, to test whether the deadline for the channel N is less than (more urgent) than the deadline for the channel M decode in progress. If so, YES, then operations branch to a step 2241 to execute a preemption process for the frame in Channel N which decodes it and expends decode execution time EN. This branch corresponds to the FIG. 21 upward arrow PACKET 3 SAVED and action of LAUNCH 3.

[0180] In step 2235, if the result is NO, then operations proceed to a step 2251 to add Frame in Channel N to the Sequence Queue (egr_sched_list) as in FIG. 13 step 1215, and block 1431 of FIGS. 14 and 16. In other words, channel N is not as urgent as channel M decode.

[0181] In step 2231, if the result is NO (either or both of First Condition and Second Condition is not true), then operations branch to a step 2261, to discard the packet in Channel N. This packet is too late to be saved, or too late to be saved without losing already commenced decode in channel M. Thus. a packet triage is executed.

[0182] After any of the steps 2241, 2251 and 2261, operations pass to RETURN 2271, whereupon the decoder in step 2211 suitably resumes any interrupted operation therein.

[0183] In FIG. 23, another embodiment has egress buffers 1175.a, 1175.b, ... 1175.q (compare FIG. 11) for depacketized data frames. The egress buffers 1175.a, 1175.b, ... 1175.q are provided with respective counters 2305.a, 2305.b,...2305.q arranged to hold counts representing or proportional to a number of bytes or words yet to process in the egress buffer 1175.i for each channel i. These counters 2305.a, 2305.b,...2305.q are counting continually during the decode process and are advantageously used as timers as follows. If 320 words need to be processed in 500 microseconds, and the counter points to 180 words left, then elapsed time T is 250 microseconds (180/320, T=(320-C) /320 and C=180).

[0184] Thus, the embodiment of FIG. 23 advantageously provides an environment (in hardware, software or mixture thereof as skilled worker chooses) to make egress data counters meaningful in measuring time. If the system is arranged so that all deadlines for decode fall on the same set of 10msec boundaries then preemption of Late packets by Very Late packets suffices to sav Very Late packets. (The system is provided with sufficient processing power to process all Very Late packets even if all channels are active and so, the order of decode execution of the Very Late packets is

less important) However, in other embodiments, the system has the deadlines for decode falling on different deadline instants other than just the 10 millisecond times. In such latter embodiments, one Very Late packet that has just arrived can and should be made to preempt another Very Late packet in decode execution because in such case the order of decode execution does matter to advantageously save the arriving Very Late Launch decode execution of the packets in order of their deadlines for launch (D_i-E_i) and preempt with suddenly arriving packets that are even closer to launch deadline. See FIGS. 21 and 23

[0185] Further in FIG. 23 operations commence at BEGIN 2321 and proceed to a step 2325 to compute a deadline interval DI for a just arrived packet in channel N corresponding to one of the egress buffers 1175.N. In the meantime a decode is taking place in a channel M. To begin a process of determining whether the decode in progress for channel M needs to be interrupted a step 2329 inputs a counter value C in counter 2305.M. Next a step 2331 tests to determine whether the just-arrived packet must be triaged, by subtracting in a First Test from deadline interval DI for packet N the amount of time C delta t needed to finish the rest of decoding in channel M. ("delta-t" is the amount of time each counter unit of C represents thereby to put DI and C on same time units.) If the subtractive difference thus determined is less than zero there is no time to spare from the decode in channel M in progress, whereupon operations branch to a step 2361 to discard the just arrived packet of channel N. Also in step 2331, a Second Test determines whether deadline interval DI is less than the execution time EN needed to execute a decode for just-arrived packet in channel N. If either the First Test or Second Test is met (YES), then triage of the just-arrived channel N packet occurs by branch to the discard step 2361.

[0186] If in step 2331 neither the First Test nor Second Test is met (NO), then operations proceed to another decision step 2335 to test whether the deadline DN for the just-arrived packet is less (comes sooner) than the deadline DM for the packet in decode in channel M. If so, then operations branch to a step 2341 to initiate a high priority interrupt routine, to preempt the decode in channel M and execute a decode for the just arrived packet in channel N, whereupon RETURN 2371 is reached and decode in channel M resumes.

[0187] If the decision step 2335 determines that the deadline DM is earlier than the deadline DN for the just-arrived packet, then operations proceed to a step 2351 to add the just-arrived packet to link list of FIG. 16 whereupon it is advantageously decoded when its turn comes. Operations at each of steps 2351 and 2361 finally reach RETURN 2371.

[0188] A block of steps 2281 in FIG. 22 and a block of steps 2381 of FIG. 23, and a block of steps 2481 of FIG. 24 provide different alternatives which may be evaluated and interchanged with one another by the skilled worker in various embodiments.

[0189] FIG. 24 shows a layer of optimization for preemption in a VoP/VoIP process earlier designated in FIG. 20 as embodiment #3. Here egress is assigned a higher priority than ingress. If a very late egress packet arrives and its deadline is below a threshold amount such as only 400 microseconds to go, then at that threshold a preemption process advantageously saves the packet from loss. The threshold amount is the length of time remaining to complete the task which is currently running on the system. The threshold amount will vary with identity of the particular task and thus be looked-up or signaled. The threshold amount varies and is calculable from the degree of completion of a particular task which has begun running according to information native to the task and for which a corresponding threshold calculation process is established according to this preemption embodiment.

[0190] In FIG. 24, the information is used in the calculation that determines whether the packet will be entered into the link list 1631 or not, i.e. just before the link list entry decision. Now in the process the process is divided into periods thereby to retain the nonpreemptive environment, waits until the break in the process occurs, and at that break time determines whether to open up the egress frame. During the break time, a control signal is passed back to a system control block for determination to schedule or drop a packet.

[0191] In FIG. 24 in measuring time with fine resolution, the arrival of encoded voice samples themselves which occupy 125 microseconds in sample period at 8KHz rate, in itself constitutes a reliable clock. When using a fine resolution embodiment, use the very same clock contemplated there as a clock that provides the decode-elapsed time as well. The skilled worker thus makes the appropriate arrangements given the hardware being improved.

[0192] The FIG. 24 embodiment uses a voice decode process 2403 itself as a timer of its own progress through decode. If the decoder process 2403 has to process 160 samples of data, the process itself every 20 samples or so may be arranged to open itself up to be interrupted. At the same time, the process every 20 samples or so can set up a register 2414 and signal to it that it has reached 20, 40, 60, etc. samples. Then the register 2414 is accessed by a scheduler 2413 for that information stored therein for use in determining how to schedule packets, and whether to preempt. In this way, decode processes of a variety types exemplified by decode process 2403 are arranged to create signals which can be used for the advantages and new processes contemplated herein.

[0193] In FIG. 24, operations commence with BEGIN 2405 and proceeds to a step 2407 to determine whether the next packet frame (e.g., in channel M) represents voice or silence. If voice (YES), then operations branch to the Decode Process 2403. There operations first initialize a register 2414 and then transfer and decode at a step 2409.1 illustratively 20 bits from a frame of voice (or media) data in an egress packet queue 1175.i of FIG. 11, or the Sequence Queue and Management block 1431 of FIGS. 14 and 16. After step 2409.1, a break process 2411.1 initiates a call to scheduler

2413 (shown as an oval above decode process 2403, and in more detail at the right in FIG. 24. Also, break process 2411.1 updates (either incrementing or decrementing methods suffice) a register 2414 so that the contents of register 2414 indicate a value proportional to or representative of the time elapsed into the decode execution of decode process 2403.

[0194] A quick pass through scheduler 2413 process steps at right in FIG. 24 takes care of any just-arrived packets, whereupon operations return to decode process 2403, and execute decode of 20 more bits at step 2409.2, followed by another break process 2411.2, and so on through 2409.3, 2411.3,...240.9k, 2411.k and decode completion at RETURN 2415.

[0195] When, as noted above, scheduler 2413 is called, operations commence at BEGIN 2421 and then a step 2425 calculates the deadline interval for a just-arrived packet in, say, channel N. Next a step 2429 inputs a value T from register 2414 representing elapsed time into decode process 2403. Operations proceed to a block of steps 2481 including steps 2431, 2435, 2441, 2451, and 2461 which are described in the same way as were the steps 2231, 2235, 2241, 2251 and 2261 in block 2281 of FIG. 22. After operations in block 2481 of FIG. 24, a RETURN 2471 is reached.

[0196] In FIG. 24 at step 2407, when the frame encountered represents an interval of silence, operations go to a step 2416 to load register 2414 with an update representing the duration of the silence. Next a step 2417 decodes the silence as no-sound or as white noise, or as decaying energy, or otherwise according to the silence decode process desired and selected by the skilled worker. A just-arriving packet in another channel suitably interrupts this silence decode process 2417 using scheduler 2413. After silence decode step 2417 operations there reach RETURN 2419.

[0197] FIG. 25 illustrates a break process used in blocks 2411.1, 2411.2, 2411.3,...2411.i of FIG. 24. Operations commence at BEGIN 2511 and then at a step 2515 save decode parameters and data from the previous 20 BITS step 2409.i to the extent needed to initiate the next 20 BITS step 2409.i+1 when the latter's time comes. After step 2515 operations at a step 2521 update register 2414 with decode execution time remaining (or decode time elapsed) as coordinated with steps 2429 and 2431 of FIG. 24. Thus, decode time elapsed is input as time T in step 2429 and used as shown in step 2431. Decode time remaining (Tr) is alternatively fed to step 2492 and the test in step 2431 is made to be [(DIN-EN>0) AND (DIM -Tr - EN >0)].

[0198] In FIG. 25 operations go from step 2521 to a step 2525 that calls the scheduler 2413 that is suitably implemented as shown in FIGS. 22, 23 and/or 24. FIG. 25 also suggests a process embodiment wherein flag software is used in the scheduler 2413. The flag is turned on when a suitable test determines that any other packet or channel should be given priority over the decode process underway in FIG. 24. When this flag is on (YES) operations branch from step 2531 to a step 2535 to start decode of the priority channel. Note that operations may nested such that even that priority decode may be interrupted, whereby starting in step 2535 does not signify uninterrupted run to completion. After step 2535, or when the bypass flag is OFF in step 2531 (NO), then operations go to a step 2541 to recover the decode parameters and data from step 2409.i, whereupon operations proceed to a step 2551 to return and go to the now-current channel decode process 2409.i+1.

[0199] FIG. 26 details a process example of embodiment type #5 of FIG. 20. Operations commence with Egress Begin 2611, and then at a step 2615 put an egress packet in a queue 2625. Next, a decision step 2621 tests to see whether Deadline Interval DI, for the latest packet in the queue 2625, exceeds a certain threshold number K of bytes (words or frames, etc. as skilled worker chooses). For example, suppose a frame 2531 is the packet that is the latest (lowest DI of all frames) in the queue. The frame 2531 is an early frame. Its DI exceeds the number K. Step 2621 determines YES and branches to step 2641, which calls an ingress process because there is plenty of time to fit the ingress process before any more egress processing is necessary in the channel.

[0200] If the test in step 2621 turns out to be NO, operations go to step 2651 to execute the egress process as discussed earlier hereinabove, whence a RETURN 2655 is reached. Also, if during the ingress process called in step 2641, the lowest deadline interval DI becomes less than K in decision step 2671, such as by entry of a newly-arrived packet 2661 into a queue 2625 position 2665, then a branch from decision step 2671 goes to egress step 2651. Otherwise, step 2671 determines YES and operations pass to RETURN From Ingress 2645 and back to step 2621.

SILENCE PROCESSING EMBODIMENT

[0201] Since silence processing has a much shorter processing duration than voice processing of a frame, the decoder in FIG. 24 suitably determines whether the packet is voice or not in block 2725. If not voice (i.e. silence), then the register 2418 is loaded with a silence processing period T = 5 microseconds, for instance, and the silence packet is decoded, followed by a return. The scheduler 2413 checks the register 2418. A very late arriving packet that might have been discarded at step 2431, is now advantageously saved because the T value is so much less at stop 2429 for a silence packet.

SINGLE CHANNEL EMBODIMENT

[0202] In a cellular phone embodiment of FIG. 4, ingress packets at the sending phone become egress packets at a remote receiving telephone. Another process organizes the scheduling process to "mistreat" or delay the ingress packets to the least possible extent. Instead of assigning the egress packets priority over everything else, the process with advantageous flexibility utilizes information about the deadline for execution of each egress packet. In addition to concentrating on those packets which are late, the process considers those packets that are early with known long deadline intervals. The process nonpreemptively holds off processing a packet which is early because it is known to be early. One or more ingress packets are thus given priority nonpreemptively and processed instead of the known-early egress packet. The ingress packets thus processed are then sent out and arrive earlier than they otherwise would have arrived at a receiving location, and thus their probability of becoming lost packets is reduced significantly. (If in a trivial situation the egress packet that was early and did not need to be processed and has nothing ahead of it, then the process simply executes on that egress packet because it is at the top of the queue.) In this way, advantageous process embodiments improve even single-channel mobile terminals, cellular telephone handsets, and other single-channel applications.

[0203] In FIGS. 4 and 26, a non-preemption scheduler sees there is time to slip an ingress encode in an open opportunity time interval because the egress channel has only known-early packet content which allows. The egress packet is looked up for deadline interval, the process sees it has lots of time left, i.e. detects a deadline interval greater than a predetermined value. Then it processes an ingress packet. Advantageously, this embodiment increases the transmission quality of the network because it confers less delay of the ingress packets in terms of their deadline interval at the receiving end.

[0204] Operations begin at Egress begin step 2611 and proceed in a step 2615 to put a new packet in the queue. Next a step 2621 determines whether the deadline interval DI exceeds a predetermined time interval K for the packet in the queue 2625 having the lowest value of DI 2631. Time interval K is at least as large as a predetermined amount of time needed for an ingress process to encode more input speech at the sending end. If in step 2621 lowest DI does exceed K, then operations branch to a step 2641 to call the ingress process which then executes in less than time interval K and returns in a step 3145 to the step 3121. At some point step 2621 detects that the lowest DI packet has aged to the point where DI is not greater than K, and operations proceed to a step 2651 to execute the egress process, whereupon Return 2655 is reached.

[0205] Since a late arriving packet 2661 may come into the picture, its arrival interrupts or preempts the ingress process if DI for the late arriving packet is determined to be not greater than K in a step 2671 executed during the ingress process begun at step 2641. If this happens, then the ingress computations are suitably saved, if they have proceeded far enough to produce meaningful ingress information, and then operations go to step 2651 to execute the egress process for the late-arriving packet 2661.

[0206] In this way, as shown in FIG. 4, ingress processing at a source 481 is advanced, thereby giving the network ample time, or at least more time, to get ingress packets to a cell phone 491 where egress processing in phone 491 assigns a higher deadline interval DI value to the incoming packet than would have otherwise been the case.

[0207] Note further that since silence processing has a shorter processing duration than voice processing, the character of silence or voice is used in determining whether to allow the ingress process to go forward in lieu of the egress process.

[0208] Advantages conferred by the system of FIG. 4 thus are 1) less delay of ingress packets, and 2) increased resulting network transmission quality.

[0209] Alternatively to FIG. 14, as shown in FIG. 27 the system initially does not differentiate between silence and voice packets 2711. Such system goes through the decode process as if the silence were a voice packet. The silence or voice packet simply go to the sequence queue management 1431. Eventually they both get picked up and go into the decode processing 2725 that simply only at this point decides whether it is a voice packet or silence packet.

[0210] In FIG. 14, the silence process bypasses decode processes 1425, whereas in FIG. 27 the silence information goes to the decode processing block 2725. In FIG. 27 the local buffer of FIG. 14 is not used, and decode postprocessing 1441 of FIG. 14 is simplified because decode processing 2725 directly puts silence or voice data directly into an output buffer 2751. Simplified postprocessing 2741 updates channel record 1413. Whether voice or silence packet is involved, the deadline interval update process of simplified postprocessing 2741 is the same. Even when the sequence queue and management process 1431 may have a little more work to do, compared to FIG. 14 because silence packets go to process 1431 as well as voice, both types of packets are advantageously handled in a uniform way all the way until decoding process 2725 occurs. Thus, selector process 1405 of FIG. 14 is obviated. Also, many processes desirably have a continuity of execution, such as adaptive filtering and echo cancellation, and G.726 encode with delta modulation change voice to differences in amplitudes and mapping of a given fragment of speech.

[0211] If detection of silenc occurs in decoding process 2725, the silence frame is suitably decoded into a ramp with a certain spectrum so that the silence is not as abrupt.

[0212] In the system of FIG. 27, the silence packets and their corresponding calls are used to schedule link list records in queue 1431 of FIG. 16. A silence packet has a usual header but a shortened payload compared to a voice packet. The payloads are stored in memory that in some embodiments have fixed locations even when the decode processing 2725 decodes from that fixed location. The code includes the capability of informing the front end queue management that a space has become available, so that when the packet has been decoded, that space is returned to availability as part of the queue space for new packets.

[0213] In both the systems of FIG. 14 and FIG. 27, the process is robust even if a silence packet and a consecutive voice packet arrive out of order. Sequence numbers of the packets are checked and deadline interval determined, thus automatically providing resequencing as the link list 1431 is updated. Then the decode knows it is a silence packet or voice packet because the header provides this information.

[0214] A management method by which the scheduling list 1431 is managed, decrements values in packets, adds packets, takes packets at top of queue and sends them to decoder. If the packet at top of queue is a silence packet, it likewise goes to the decoder 2725 in the FIG. 27 embodiment. The decoder detects that it is a silence packet, and detects the number of words of silence. Then the postprocessor 2741 updates the channel record with updated deadline interval. Silence processing takes less time (e.g., 5 microseconds) than voice processing, at say, a few hundred microseconds. Silence processing suitably fills an area of output buffer 2751 or memory with zeroes or noise with predetermined spectrum similar to previous voice packet, or interpolation, or other silence data.

[0215] Since silence processing has a different processing duration, the decoder or a data store takes account of this in operating the processor to consider it as a very late packet, or to issue timing to a register; or in determining whether to allow the ingress process to go forward in lieu of the egress process as described earlier above.

[0216] The decoder tells the scheduling list how far it is in its processing. In some embodiments the decoder and scheduler both form part of a nonpreemptive scheduling loop.

[0217] In preemptive embodiments, the process suitably is made to look ahead where a segmented decoder process of FIG. 24 allows itself to be interrupted because of a late arriving packet.

[0218] Conversely, suppose a silence packet is the arriving packet. Even if the silence packet is very late, even later than a voice packet which would have been discarded, the silence packet also has a larger deadline interval DI because it takes less time to be processed. In some embodiments a lost packet is treated the same as a received silence packet, and this consideration is less important. But in other embodiments a received silence packet is decoded to produce a different output sound from the sound which the decoder produces in response to a lost packet event detected in the last 10 milliseconds before the deadline for example. Also, in embodiments where a received silence packet contains a number indicating a value S of how many frames of silence, and a lost packet event only signifies a predetermined number of lost voice frames, then saving a silence packet from loss is also important. Thus, the deadline interval calculation advantageously saves silence packets for decoding that would otherwise be discarded.

CIRCULAR OPERATIONS

[0219] The next sections describe a solution to the problem of handling packet processing deadlines in the context of a system having circular buffers and a system timer counter rollover (circular time) and packet time-stamp rollover.

Counter Size

[0220] To use circular time, select a desirable clock rate r_{CLK} [Hz], and corresponding clock cycle period $c_{CLK}=1/r_{CLK}$, and an estimate of the largest interval of time t_{MAX} [sec] that ever needs to be maintained by the system. That provides the required clock range [number k] which leads to number of bits [number b] needed to implement the clock. The size T_{MAX} of the system clock (clock range) is made at least twice the largest time span t_{MAX} between any two most time distant packets that will ever occur in the system.

[0221] For standard 8kHz telephony, a 16-bit clock provides a total span of over 8 seconds ($125\mu sec*2^{16}>8sec$) which substantially exceeds the longest time span between any traces of two most distant streaming media packets that will likely exist concurrently in a given channel in a global system. Thus $t_{MAX}=2^{16}$ and $k-1 = 2^{16}-1$.

[0222] Let the 16-bit circular time clock have a buffer of the size 2^{16} words having 2^{16} addresses.

[0223] To create an "automated" process, use a system clock modulus T_{MAX} where

$$T_{MAX} = 2*t_{MAX};$$

where t_{MAX} is defined as before: t_{MAX} is the linear time range or difference that includes any two most distant events, the record of which is ever concurrently present in the system.

[0224] Define now new operation $circ_T(t_A, t_B, T_{MAX})$ for any two values t_A and t_B where $0 \le t_A < t_{MAX}$; $0 \le t_B \le t_{MAX}$. Let

$$\Delta = t_A - t_B ;$$

then

if $0 \leq \Delta < t_{MAX}$ then $circ_T (t_A, t_B) = \Delta$; [$circ_T$ is positive, no straddle] CASE 1

if $-t_{MAX} \geq \Delta > -T_{MAX}$ then $circ_T (t_A, t_B) = \Delta + T_{MAX}$;

[$circ_T$ is positive, straddling] CASE 2

if $0 > \Delta > -t_{MAX}$ then $circ_T (t_A, t_B) = \Delta$; [$circ_T$ is negative, no straddle] CASE 3

if $t_{MAX} \leq \Delta < T_{MAX}$ then $circ_T (t_A, t_B) = \Delta - T_{MAX}$;

[$circ_T$ is negative, straddling] CASE 4

If after obtaining value of Δ as a result of above

If Δ equals T_{MAX} then $circ_T(t_A, t_B) = 0$
If Δ equals $-T_{MAX}$ then $circ_T (t_A, t_B) = 0$

[0225] FIG. 28 illustrates the four results above designated Case 1, Case 2, Case 3, and Case 4. Rightward arrows for Δ or $circ_T$ indicate that their sign is positive, wherein event A has occurred after event B. results are designated Case 1, Case 2, Case 3, and Case 4 in FIG. 28. Leftward arrows for Δ or $circ_T$ indicate that their sign is negative, wherein event A has occurred before event B.

[0226] A clock face 2811 shows circular time that goes from one (1) "o'clock" to T_{MAX} as its "twelve o'clock." Times t_A and t_B show up as large dots on the clock face 2811. A 180-degree turn is designated t_{MAX}. This 180-degrees is a span within which the system determines the order of events A and B using the circular time difference process described here. A small circle 2821 suggests the system clock incrementing with fine resolution like a second hand on clock face 2811, as linear time moves from left to right from zero (0) to twice T_{MAX}.

[0227] In FIG. 29 circular time operations to provide circular time difference Δt_θ, which signifies $circ_T (t_A, t_B)$, commence with a BEGIN 2905 and proceed to input tags or computed values representing a pair of times t_A, t_B. Next a step 2911 finds their linear time difference $\Delta = t_A - t_B$. A decision step 2915 tests the sign bit of the difference from step 2911 next. If the sign bit is positive, then a decision step 2917 tests the most significant bit (MSB) of the difference from step 2911.

[0228] If the MSB is zero (0), meaning the difference is in the low half of an interval zero to + T_{MAX}, then a Case 1 condition of $0 \leq \Delta < t_{MAX}$ is present and Δt_θ is set equal to Δ. Then a step 2925 sets a flag representing that Δt_θ is positive. If the MSB is one (1) in step 2917, meaning the difference is in the high half of an interval zero to + T_{MAX}, then a Case 4 condition of $t_{MAX} \leq \Delta < T_{MAX}$ is present and Δt_θ is set equal to $\Delta - T_{MAX}$ .in a step 2931. Then a step 2935 sets a flag signifying that the circular time difference Δt_θ is negative. Note this advantageously remarkable operation: a POSITIVE LINEAR time difference is converted and interpreted to be a NEGATIVE CIRCULAR time difference sometimes and not others.

[0229] Turning now to the other side of decision step 2915, suppose that the sign bit is negative for the linear time difference $\Delta = t_A - t_B$. Then a decision step 2941 tests the most significant bit (MSB) of the difference from step 2911.

[0230] If the MSB is one (1), meaning the difference is in the low magnitude half of an interval zero to - T_{MAX}, then a Case 3 condition of $0 > \Delta > -t_{MAX}$ is present and Δt_θ is set equal to Δ in a step 2945. Then operations go to the step 2935 which sets a flag signifying that the circular time difference Δt_θ is negative. If the MSB is zero (0) in step 2941, meaning the difference is in the high magnitude half of an interval zero to - T_{MAX} , then a Case 2 condition of $-t_{MAX} \geq \Delta > -T_{MAX}$ is present and Δt_θ is set equal to

[0231] $\Delta + T_{MAX}$ in a step 2951. Then operations go to the step 2925 which sets a flag representing that Δt_θ is positive. Note this advantageously remarkable operation: a NEGATIVE LINEAR time difference is converted and interpreted to be a POSITIVE CIRCULAR time difference sometimes and not others.

[0232] After either of steps 2925 and 2935 operations then go to RETURN 2927, completing the operations of this advantageous circular time differencing embodiment.

[0233] The four components of the process above are ordered according to the results. Less procedurally complex is the view ordered according to the value of Δ. This approach is illustrated in FIG. 30.

if

$$(\Delta \geq 0)$$

{ if ($\Delta < \frac{1}{2}T_{MAX}$) $\Delta t_\theta = \Delta$; else $\Delta t_\theta = \Delta - T_{MAX}$; } else
{ if ($\Delta \geq -\frac{1}{2}T_{MAX}$) $\Delta t_\theta = \Delta$; else $\Delta t_\theta = \Delta + T_{MAX}$; }; where

$$\Delta t_\theta = circ_T\ (t_A, t_B, T_{MAX}).$$

Thus, in FIG. 30, the high half intervals appear at the extreme left and extreme right quarters of the line or axis representing values of linear time difference delta. In those two extreme quarters, an operation of ADD 3011 or SUBTRACT 3021 operates on bits representing linear time difference delta, which bits are temporarily stored in a register 3031. Further in FIG. 30, notations for signs and MSBs are entered for each quarter of the line.
Here was used ≥ while could use > for the sake of the clarity of the next section "Looking at the Sign Bits".

[0234] The above statements replaced the values of t_{MAX} with that of $\frac{1}{2}*T_{MAX}$. This is permissible both here as it would be in the quad above, since by definition $t_{MAX} \leq \frac{1}{2}*T_{MAX}$. Thus the inequalities hold up correctly.

[0235] Also eliminated was the comparison with T_{MAX}. Because of circularity all values are within the $-T_{MAX}$, $+T_{MAX}$ range and no comparisons are needed.

Looking at the Sign Bits

[0236] Looking at the single statement solution in "Closing statements, section 2" above, notice that

- "if ($\Delta \geq 0$)" is a test of the sign of Δ.
  The sign in 2-complement arithmetic, now universally adopted in DSP computers, see FIG. 29, is encoded as the leftmost bit. It is "0" for zero and positive values, and "1" for the negative values. Just inside the test of the sign there is another test.
- "if ($\Delta < \frac{1}{2}T_{MAX}$)" is a test of the most significant mantissa bit
  Consider that Δ is positive. In a 3-bit mantissa arithmetic 000, 001, 010 and 011 (0,1,2 and 3) are all less then half of full range, while 100, 101, 110 and 111 (4,5,6 and 7) are greater then half of full range. Thus the decisive factor (Fig. 29) is the most significant bit (MSB) of value. If it is 0, the value is less than the half of T_{MAX}. Otherwise it is greater. Now consider Δ being negative. Inside this side of the statement there is the test
- if ($\Delta \geq -\frac{1}{2}T_{MAX}$)
  Again enumerate three bit mantissas, but this time in negative numbers: 111, 110, 101, 100 (-1,-2,-3,-4) have the most significant bit (MSB) equal "1", while 011, 010, 001, 000 (-5,-6.-7,-8) that is the larger negative numbers have "0". Again. the decisive factor in FIGS. 29, 30 and 32 is the MSB, or most significant bit of value.

Hardware or hardware-like solution

[0237] In FIG. 31, a semiconductor chip embodiment has a core microprocessor, microcontroller, or digital signal processor 3110 combined on a single chip with a section of nonvolatile memory 3112 and sections 3113 and 3115 of SRAM (static random access memory). The nonvolatile memory 3112 is loaded with, or manufactured to have stored therein, the software of FIGS. 1, 5 and 6, and FIGS. 9 through 29, such as blocks of speech coder, packetize, VoIP control, IP stack, and GUI as the skilled worker selects. Further in FIG. 31, the single-chip integrated circuit DSP 3110 has an instruction decoder 3117, at least one ALU (arithmetic/logic unit) 3131 and a multiplier unit 3121. Buses 3141 interconnect decoder 3117, ALU 3131, multiplier 3121, and memories 3112, 3113, 3115, with a DMA and bus interface unit 3151.

[0238] In FIG. 32, the comparison $\Delta = circ_T(t_A , t_B)$ produces real values of the time intervals. In reality, sorting and purging processes of FIG. 13 are concerned primarily with the sign of $circ_T$, meaning determining which of two time stamps t_A and t_B is bigger, and which is smaller.

[0239] The sign of $circ_T$ is determined from the sign of Δ and the next MSB bit.

00 implies ($t_A > t_B$)
01 implies ($t_A < t_B$)
10 implies ($t_A > t_B$)
11 implies ($t_A < t_B$)

[0240] Surprisingly, however, the outcome of the comparator-type, or sign only comparison is determined by the

second most significant bit, when a conventional arithmetic is used, and can be determined by uncomplicated logic when other arithmetics are used.

[0241] FIG. 32 shows a decision step or decision hardware element 3211. If in the register 3031, the second most significant bit of Δ (the most significant bit of value) is 0 (zero), then Δt_θ is positive which implies that t_A is arithmetically larger then t_B and therefore t_A represents a more recent event. Otherwise, t_A is an earlier event then t_B.

[0242] The methods are advantageously utilized in both software and hardware embodiments

$$if\ ((\Delta\ \&\ 0x8000) = 0)...$$

[0243] In the statement the second leftmost bit of the 16-bit Δ is isolated by marking it off, and compared to "0". Thus, if (Δ & 0 x 8000) is "True" then $t_A \geq t_B$, and $t_A < t_B$ otherwise.

[0244] Further in FIG. 32, an adder/subtractor circuit 3231 is supplied with values of times t_A and t_B at inputs thereto. The SIGN bit and the MSB bit from register 3031 are fed to a logic circuit 3235 which is designed by ordinary skill to drive an ADD/SUBTRACT control input of adder/subtractor circuit 3231 to appropriately cause an add or a subtract therein according to Cases 2 (two) and 4 (four) respectively when they occur. The result of adding (Case 2) or subtracting (Case 4) or doing nothing (Cases 1 and 3) is fed from adder/subtractor circuit 3231 to register 3031 or to memory via a bus 3241. In this way circular time difference CIRCT is implemented in software or high speed hardware for a variety of applications.

[0245] In FIG. 33, when packets are sorted by time-to-deadline (DI) to determine which is to be processed first, then advantageously carry on comparisons

$$\Delta t = circ_T\ (t_{DDL,M},\ t_{DDL,X},\ T_{MAX});$$

[0246] Interpreting the potential result, if $\Delta t > 0$ (is positive) implies that the item already on the queue has longer deadline, and the item being sorted in should be inserted before. If $\Delta t \leq 0$ the sorting process should step to the next item on the queue.

[0247] The above process 3300 advantageously and efficiently "updates" the queue 1431. Compare FIG. 33 with FIGS. 16 and 17.

[0248] Operations of process 3300 commence at a BEGIN 3305 and initialize a Queue Record pointer at a step 3311 before entering an updating loop. Then a step 3321 computes the circular time difference $\Delta t = circ_T\ (t_{DDL,M},\ t_{DDL,X},\ T_{MAX})$ between a record to be sorted into the queue and the time associated with a record already in the queue. Next a decision step 3331 determines whether the circular time difference is positive. If not ,operations at step 3341 step to the next queue record and loop back to step 3321. Finally, the sorting loop determines that the circular time difference is positive, and operations to proceed to a step 3351 to insert the new record ahead of the current queue record, thereby sorting the new record into the queue. Thereupon, a RETURN 3361 is reached.

[0249] FIG. 34 depicts a media over packet computer system having improved packet playout control software of FIGS. 1, 5, 6 and 9-29 for both speech/audio and image real-time information. A media over packet control block 3411 is interconnected with speech codec and/or audio codec 3421, a speech/audio ingress/egress software control block 3431, a packetize block 3441, depacketize block 3443, a TCP/UDP/IP stack 3451, an image compressor/decompressor 3471, and an image ingress/egress control block 3473.

[0250] A microphone and A/D circuit supply speech/audio codec 3421 with sampled audio information. Speech/audio codec 3421 supplies frames via speech/audio ingress/egress software control block 3431 and to packetize block 3441 which couples to at least one software object connecting via a modem and/or link/physical layer block 3461 to a network or wireless communications path. Block 3461 is any kind of modem or any device that has a link layer and/or physical layer for communication purposes.

[0251] Various image sources supply image data via a control interface CTRL I/F to video compressor and/or image compressor 3471. The image sources include a television Tuner, a VCR video tape recorder and player, a video camera, a CD-ROM drive having images on it, a digital still camera, the output of a medical image processing computer such as a CAT (computer aided tomography) scanner, PET (positron emission tomography) scanner, MRI (magnetic resonance imaging) scanner or other image processing computer (not shown).

[0252] In an architecturally elegant and similar way to the audio blocks above, image compressor 3471 supplies frames via image ingress functions in image ingress/egress software control block 3473 and to packetize block 3441 which couples to software object B connecting via the modem 3461 to the network. Real time data such as from a medical monitor or industrial process control (not shown) are suitably coupled analogously into the transmit path.

[0253] In the receive direction a software object B receives packets from one or more communication paths in the network and supplies them to depacketize block 3443 coupled to egress functions in the speech/audio ingress/egress

software control block 3431. Software 3431 provides depacketized frames to audio codec 3421 for decoding of frames for output through D/A converter and an audio output transducer such as a loudspeaker. Packet playout control and lost packet compensation in software 3431 are performed as described elsewhere herein.

[0254] In an architecturally elegant and similar way in the image path, the software object also receives image packets from one or more communication paths in the network and supplies them to depacketize block 3443 coupled to image ingress/egress control software 3473, which in turn provides image data playout control and lost packet reconstruction for image decompression in block 3471 for decompression of image frames. Decompressed image frames are output to a Display system such as television or cathode ray tube monitor, or liquid crystal display, or digital micromirror display or digital light processing display, video recorder, image processing equipment, storage area network (SAN), or other image utilization systems.

[0255] The media over packet control 3411 is coupled to a GUI graphical user interface 3481 associated with the Display. The GUI 3481 is suitably controlled by an infrared or Bluetooth wireless link from a remote control source 3483, from a wireless or wired keyboard 3485, and/or from a wearable communication and control device.

[0256] Real time data such as from a medical monitor or industrial process control are suitably coupled from the network via software object B analogously into the receive path in packets coupling to a medical and/or industrial information processing and display.

[0257] In gateway, wireless base station, and other applications a recoder 3491 is suitably enabled by media over packet control 3411. Image information decompressed by video/image codec block 3471 and audio information decoded by audio codec 3421 are both coupled to the recoder 3491. Recoder 3491 then recodes or transcodes the information and produces an output compressed and coded according to a different form than was received by system 3400. It is contemplated that systems such as those shown in FIGS. 1, 3, and 4-6 as well as system 3400 of FIG. 34 are suitably cascaded and integrated for various telecommunication and networking purposes. Where many channels are processed simultaneously, the systems are suitably replicated or multiplexed to the extent desired, so that software and hardware are effectively, efficiently and economically employed.

[0258] Where blocks are shown herein, they are suitably implemented in hardware, firmware or software in any combination.

[0259] FIG. 35 depicts an optically accessible storage disk 3511 that has physical variations representing bits of information. In one embodiment the bits of information represent processor instructions such as DSP instructions for speech encoder 541 and ingress/egress control 581 coupled to packet software stack DSP and MCU instructions. Instructions in the packet software stack direct packets containing the real-time information from the sender computer 203 by at least one path in the packet network 200 to the receiver computer 205.

[0260] In a further storage disk 3511 embodiment, the bits of information represent any of the operations established in FIGS. 1, 5, 6, 9-29 and 33.

[0261] In FIG. 36 storage 3611 is provided by a rotatable magnetically readable hard disk storage disk 3621 bearing any or all of the instructions described in connection with FIG. 35. The hard disk 3621 is controlled and read by a hard disk drive control circuitry assembly 3631 having a read channel 3633, microcontroller or DSP 3637, and a memory 3635 interconnected for motor control, and actuator control to read and write disk 3621 from a read write head (not shown). Storage 3611 is connected by an IDE, PCI or other suitable coupling 3641 to a computer printed circuit board or add-in card 3651. The card 3651 has a microprocessor 3653, memory 3655, DSP 3657 and modem 3659 interconnected to provide packets from and to a connector 3661.

[0262] In FIG. 37, some embodiments add memory and a control program for prestoring and playing coded speech to augment or even replace the speech codec in some appliances and talking toys that speak for themselves in normal operation or during maintenance. MCU 3731 has a VoIP control 3731 and a TCP/UDP/IP packet network protocol stack 3733 and an ingress/egress control block 3724 and sequence queue block 3726 coupling a speech encoder/decoder 3723 to a network using the software as described elsewhere herein. Busses 3725 and 3727 couple a host computer 3711 to DSP 3721 and MCU 3731 executing the aforedescribed software and further couple all of them to a modem or Link/Physical Layer 3741, analog front end AFE to microphone and loudspeaker, and to Peripherals coupling to touchpad KBD and display. In this way, advantageous media over packet is accomplished in computers, IP phones, talking toys and home appliances such as refrigerators, microwave ovens, bread machines, blenders, coffee makers, laundry machines, dryers, sweepers, thermostat assemblies, light switches, lamps, fans, drape and window shade motor controls, surveillance equipment, traffic monitoring, clocks, radios, network cameras, televisions, digital telephone answering devices, air conditioners, furnaces and central air conditioning apparatus. These and other devices are suitably connected to a packet network wirelessly or via cable, telephone lines, power lines or otherwise for remotely located monitoring, control, user commands and maintenance.

[0263] In FIG. 38, an embodiment advantageously addresses management of one or more gaps or holes in the reserves of FIG. 9 egress channel buffers. In FIG. 9 for simplicity, the reserves (shaded) were shown as a contiguous sequence of samples from a circular addressing point of view, even when the samples wrap around buffer boundaries as in FIG. 9 egress channel buffers D, E and F.

[0264] By contrast, in FIG. 38, two holes illustratively exist in the reserve in an egress channel buffer 3801. Each incoming packet has a time stamp like a number in a sequence 1,2, 3, 5,6,7. The number 4 is missing from the foregoing sequence, and represents a hole. The packet with the time stamp represented by the number 4 may arrive tardily but nevertheless in time for decode. Where the size of a reserve is used to express its priority for decode, an embodiment desirably handles reserves with holes in an intelligent manner. In one type of embodiment, the size of the reserve is still computed as the distance from beginning to end even if there are small holes. For example, software suitably starts at the smallest DI frame in the reserve and stops the count or computation as soon as a gap of a predetermined width (e.g. greater than 2 frames) is encountered. For gaps of less than or equal to the predetermined width, the count continues. In this way, a reasonable estimate of the degree to which a channel is backing up in the buffer is obtained.

[0265] The channel record 1413 of FIG. 15 is suitably enhanced with additional entries:

1) End of Egress Buffer deadline
2) Hole Pointer value
3) Switch on or off, where switch (flag) is turned on when at least one hole is present in reserve 3801, for that channel.
4) Width of hole
5) Sequence number of missing packet (if computable)

[0266] If plural holes occur, different embodiments suitably handle them. In one approach, only the hole having the most urgent deadline is tabulated in channel record 1413. In a second alternative, the two holes having the smallest deadline intervals DI are tabulated in channel record 1413. In a more complex embodiment, software tabulates all holes.

[0267] FIG. 38 shows operations for filling a hole in a process which commences at a BEGIN 3805 in response to arrival of a new packet. Next, a decision step 3811 detects whether the Hole Switch has been turned according to a monitoring process as described above. If so (YES), a next step 3821 checks the sequence number of the packet relative to the hole pointer value. Then a decision step 3831 determines if the sequence number of the packet equals the hole pointer value. If so, a step 3841 fills the hole by putting the new data from the packet, suitably decoded if need be, into the storage area that constitutes the hole. Then a step 3851 clears the Hole Switch off, whereupon a RETURN 3861 is reached. Note that if the Hole Switch was originally off in step 3811, or the packet sequence number matched no hole pointer value in step 3831, then operations branch respectively from steps 3811 and 3831 to RETURN 3861.

[0268] Thus, various embodiments shown and others as taught herein confer advantages on packet communications processes, devices and systems. Where blocks are shown herein, they are suitably implemented in hardware, firmware or software in any combination.

Claims

1. An electronic system comprising:

   a processor circuit;
   an egress packet control having stored instructions that generate for first and second received packets respective deadline intervals and order the processing in the processor circuit of the first and second received packets according to the respective deadline intervals; and
   a printed wiring assembly bearing the processor circuit and the egress packet control.

2. An electronic system as claimed in claim 1 further comprising

   telephone interface circuitry having plural connectors ready for connection to plural telephone units;
   a digital network interface; and
   an integrated circuit assembly coupling the telephone interface circuitry to the digital network interface, the integrated circuit assembly including said printed wiring assembly and providing voice over packet transmission and reception as a private branch exchange.

3. An electronic system as claimed in claim 1 further comprising

   cellular telephone wireless transmit/receive interface circuitry,
   a packet network interface; and
   an integrated circuit assembly coupling the cellular telephone wireless transmit/receive interface circuitry to the packet network interface, the integrated circuit assembly including said printing wiring assembly and pro-

viding voice over packet transmission and reception as a wireless base station.

4. An electronic system as claimed in claim 1 further comprising a single-chip integrated circuit including the processor circuit and the stored instructions of the egress packet control.

5. An electronic system as claimed in claim 1 wherein the stored instructions define temporary storing of information about the packets on a queue in order of the respective deadline intervals.

6. An electronic system as claimed in claim 5 wherein the stored instructions further define periodic decrementing of the deadline intervals as time passes.

7. An electronic system as claimed in claim 5 wherein the stored instructions further define computing for a third received packet a further deadline interval and sorting the queue to insert information about the third received packet on the queue in order of its further deadline interval relative to the respective deadline intervals already on the queue.

8. An electronic system as claimed in claim 5 wherein the stored instructions further define decoding a packet having the shortest time to deadline as expressed by its deadline interval.

9. An electronic system as claimed in claim 1 wherein the stored instructions further define generation of information about the packets in the form of primary keys comprising deadline intervals and secondary keys comprising frame sizes, and temporarily storing information about the packets on a queue in order of the primary keys, and for packets having identical primary keys storing the information about the packets in order of the secondary keys.

10. An electronic system as claimed in claim 1 wherein the egress packet control defines for an arriving packet the deadline interval DI as the difference between an arrival time A of the arriving packet and a deadline time D.

11. An electronic system as claimed in claim 1wherein the egress packet control defines the deadline interval Dj for an arriving packet j having a sequence number Sj and frame width F, from a deadline Di of an earlier packet i having a sequence number Si substantially as a function of frame width F and the difference of sequence numbers Sj and Si.

12. An electronic system as claimed in claim 1 wherein an arriving silence packet j represents a number of silence frames S having a frame width F and the egress packet control defines a deadline interval Dlj for the arriving silence packet j from a deadline interval Dli previously determined for an earlier packet i, substantially as the sum of the deadline interval Di added to a product of the number of silence frames S times the frame width F.

13. An electronic system as claimed in claim 1 wherein the egress packet control has stored instructions that:

define buffers;

compute sizes of respective information reserves in the buffers; and
order processing of the packets according to a priority depending at least in part on the sizes of the respective reserves.

14. An electronic system as claimed in claim 1 wherein the egress packet control has stored instructions to preempt processing of the first received packet by preemptively processing the second received packet, when the deadline interval for the second received packet is less than the deadline interval respective to the first received packet by a predetermined amount

15. An electronic system as claimed in claim 1 wherein the egress packet control has stored instructions for ingress packet processing, and for preempting egress packet processing of an egress packet having a deadline interval exceeding a value, by preemptively executing ingress packet processing of at least one ingress packet.

16. An electronic system as claimed in claim 1 having channel decoders that operate on non-coincident frame boundaries, and wherein if the second packet has a second deadline earlier than the first deadline, the egress packet control has stored instructions for testing to determine whether both the second and first packets can be decoded ahead of their respective deadlines and if so, then preemptively executing decode of the second packet.

17. An electronic system as claimed in claim 1 wherein the egress packet control has stored instructions for a process of generating data descriptive of circular time differences between times of events A and B, the process comprising:

electronically subtracting a value representative of the time of event B from a value representative of the time of event A, there resulting a most significant bit (MSB); and
providing the MSB itself as a flag indicating which of events A and B is prior to the other.

18. An electronic system as claimed in claim 17 wherein the stored instructions are further for:

electronically subtracting and delivering to a storage element a value representative of the time of event B from a value representative of the time of event A, resulting in an electronic representation (delta) having the most significant bit (MSB) and a sign bit S; and
electronically processing the electronic representation (delta) and a predetermined value (TMAX) in response to the MSB and the sign bit S to generate the circular time difference.

19. An electronic system as claimed in claim 1 whereas the egress packet control has stored instructions for the computing for each of said received packets respective deadline intervals as circular time differences between a respective deadline D and a respective packet arrival time A.

20. A wireless telephone comprising an antenna, a voice transducer, and at least one integrated circuit assembly coupling the voice transducer to the antenna, said at least one integrated circuit assembly providing voice over packet transmissions and having embedded electronic instructions comprising an ingress/egress packet control to process egress information and determine lowest first egress deadline interval DI and further execute an ingress process preempting the egress process when the value of lowest first egress deadline interval DI exceeds a predetermined amount K.

21. A wireless telephone comprising an antenna, a voice transducer, and at least one integrated circuit assembly coupling the voice transducer to the antenna, said at least one integrated circuit assembly adapted to generate data descriptive of circular time differences between times of events A and B by electronically subtracting a value representative of the time of event B from a value representative of the time of event A, there resulting a most significant bit (MSB); and providing the MSB itself as a flag indicating which of events A and B is prior to the other.

22. A wireless telephone as claimed in claim 21 wherein the integrated circuit assembly is adapted to generate circular time differences between times of events A and B by electronically subtracting and delivering to a storage element a value representative of the time of event B from a value representative of the time of event A, resulting in an electronic representation (delta) having a most significant bit (MSB) and a sign bit S; and electronically process the electronic representation (delta) and a predetermined value (TMAX) in response to the MSB and the sign bit S to generate the circular time difference.

23. A single-chip integrated circuit comprising:

storage for values representative of the time of two events;
an adder/subtractor coupled to the storage and operative to generate a difference value (delta) and deliver the difference value into said storage thereby resulting a most significant bit (MSB) of the difference value (delta); and
a flag register having a bit fed with said MSB to indicating which of events A and B is prior to the other.

24. A single-chip integrated circuit as claimed in claim 23 further having:
logic circuitry responsive to the MSB and a sign bit S of the electronic difference (delta) and a predetermined value (TMAX), the logic circuitry for driving said adder/subtractor to generate the circular time difference of the two events.

25. A method of processing first and second received packets of real-time information,
comprising the steps of:

computing for each of said received packets respective deadline intervals; and
ordering processing of the first and second received packets according to the respective deadline intervals.

26. A process of generating data descriptive of circular time differences between times of events A and B, the process

comprising:

electronically subtracting a value representative of the time of event B from a value representative of the time of event A, there resulting a most significant bit (MSB); and
providing the MSB itself as a flag indicating which of events A and B is prior to the other.

27. A process of generating circular time differences as claimed in claim 26 further comprising:

electronically delivering to a storage element a value representative of the time of event B subtracted from a value representative of the time of event A, resulting in an electronic representation (delta) having the most significant bit (MSB) and a sign bit S; and
electronically processing the electronic representation (delta) and a predetermined value (TMAX) in response to the MSB and the sign bit S to generate the circular time difference.

28. A single chip integrated circuit comprising:

a processor circuit; and
an egress packet control having stored instructions that generate for first and second received packets respective deadline intervals and order the processing in the processor circuit of the first and second received packets according to the respective deadline intervals.

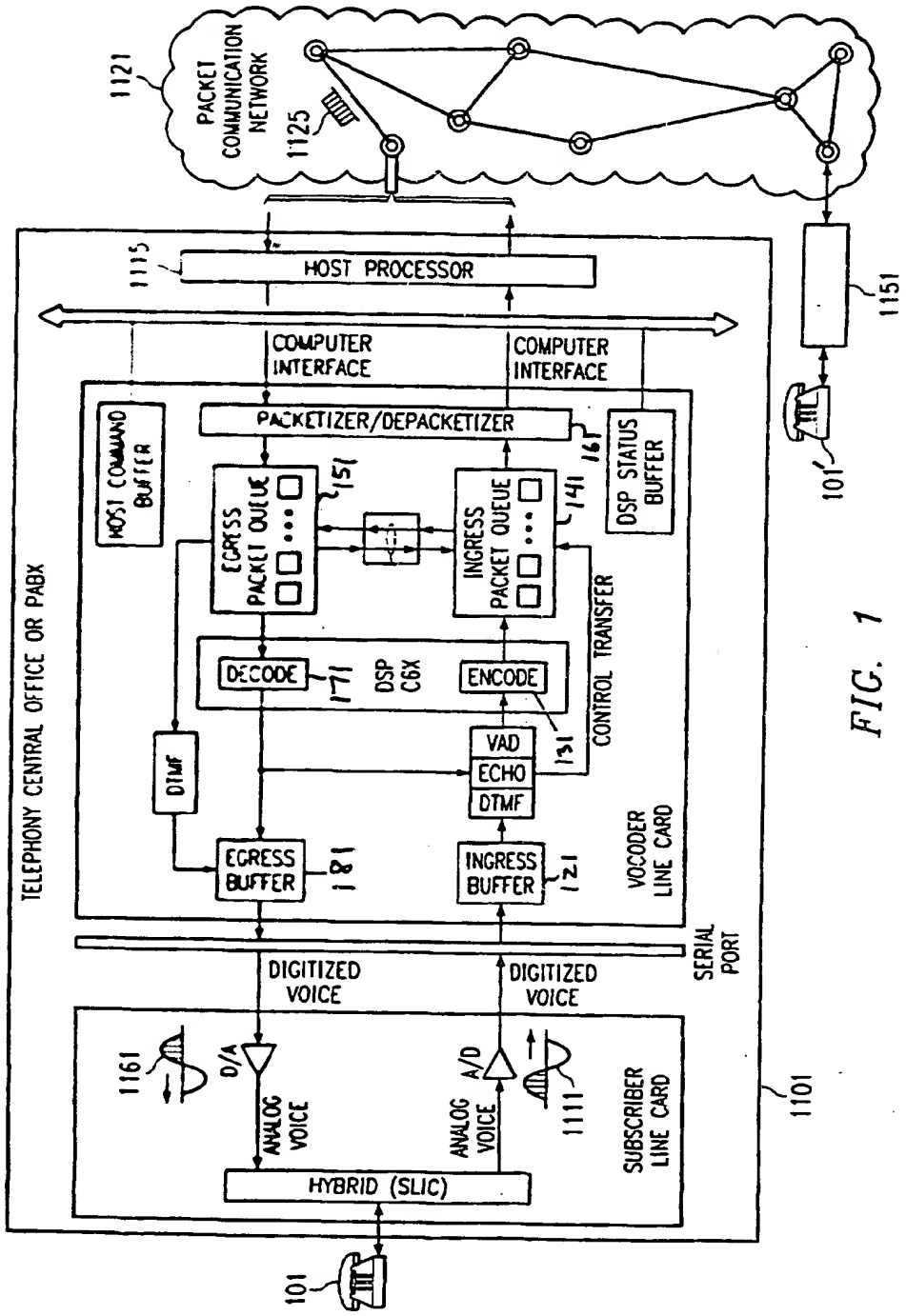

1121
PACKET COMMUNICATION NETWORK
1125
1115
HOST PROCESSOR
1151
COMPUTER INTERFACE
COMPUTER INTERFACE
PACKETIZER/DEPACKETIZER
167
HOST COMMAND BUFFER
DSP STATUS BUFFER
EGRESS PACKET QUEUE
151
INGRESS PACKET QUEUE
141
CONTROL TRANSFER
DECODE
171
DSP C6X
ENCODE
131
VAD
ECHO
DTMF
DTMF
EGRESS BUFFER
181
INGRESS BUFFER
121
VOCODER LINE CARD
SERIAL PORT
DIGITIZED VOICE
DIGITIZED VOICE
1161
D/A
A/D
1111
ANALOG VOICE
ANALOG VOICE
SUBSCRIBER LINE CARD
HYBRID (SLIC)
101
101
1101
TELEPHONY CENTRAL OFFICE OR PABX

FIG. 1

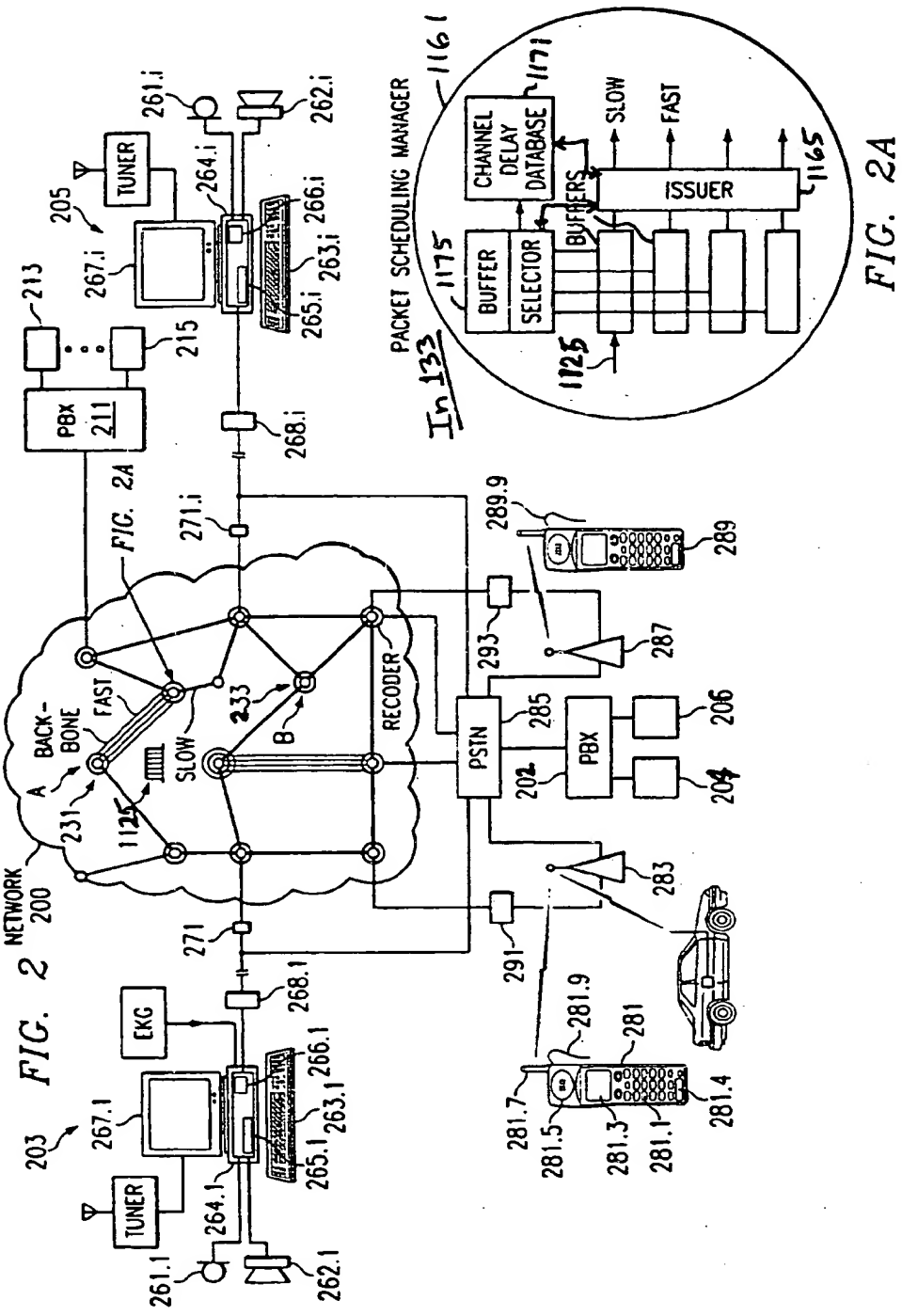

NETWORK 200
BACK-BONE
A
231
FAST
SLOW
1125
233
B
RECODER
PSTN
202
PBX
285
206
204
293
287
289.9
289
271.i
268.i
213
215
PBX 211
205
TUNER
267.i
264.i
261.i
262.i
263.i
265.i
266.i
271
268.1
EKG
203
267.1
263.1
265.1
266.1
264.1
TUNER
261.1
262.1
291
283
281.9
281
281.7
281.5
281.3
281.1
281.4
PACKET SCHEDULING MANAGER
1161
In 133
CHANNEL DELAY DATABASE
1171
SLOW
FAST
ISSUER
1165
BUFFERS
BUFFER
SELECTOR
1175
1125

FIG. 2

FIG. 2A

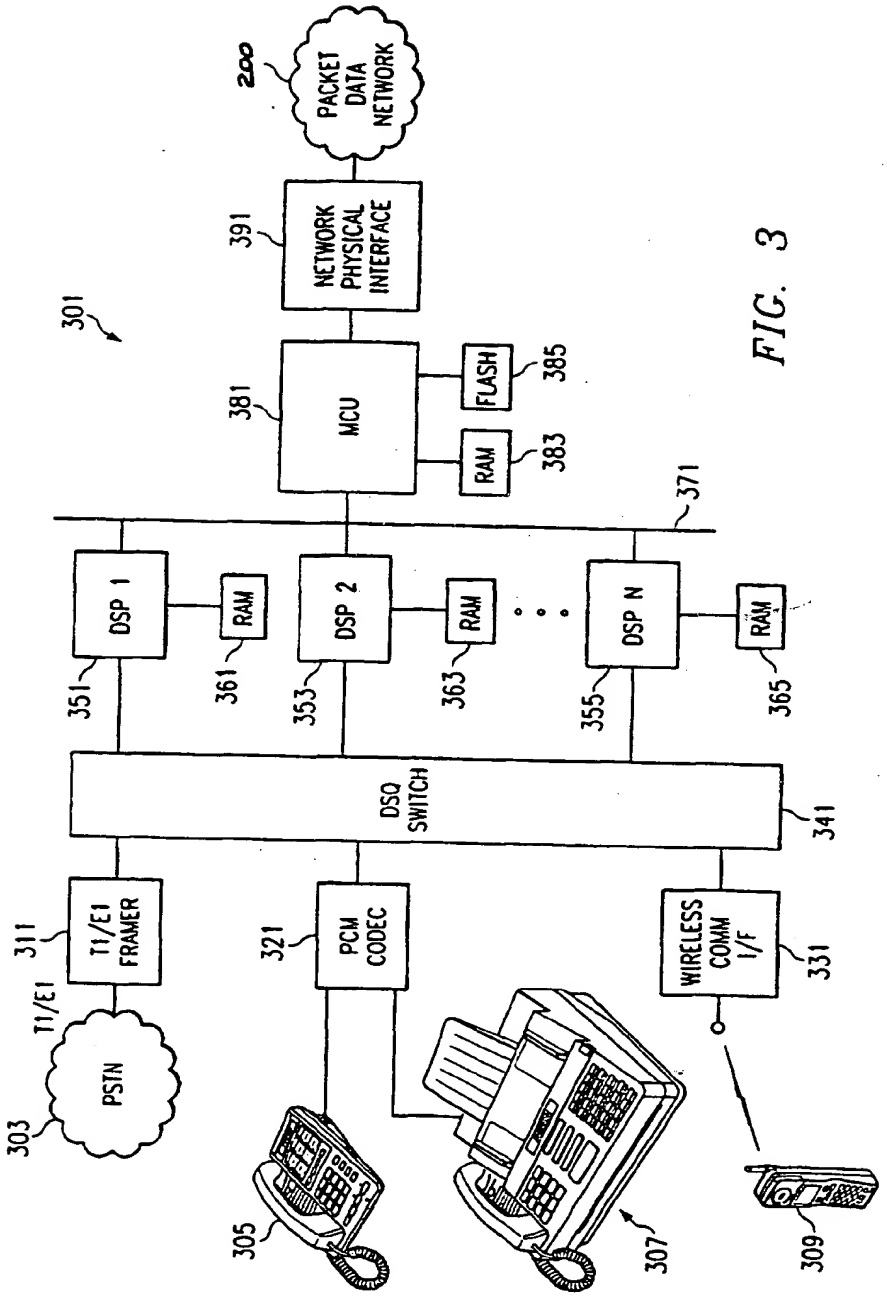

200
PACKET DATA NETWORK
391
NETWORK PHYSICAL INTERFACE
301
381
MCU
FLASH
385
RAM
383
371
DSP 1
RAM
DSP 2
RAM
DSP N
RAM
351
361
353
363
355
365
DSQ SWITCH
341
311
T1/E1 FRAMER
T1/E1
321
PCM CODEC
WIRELESS COMM I/F
331
PSTN
303
305
307
309

FIG. 3

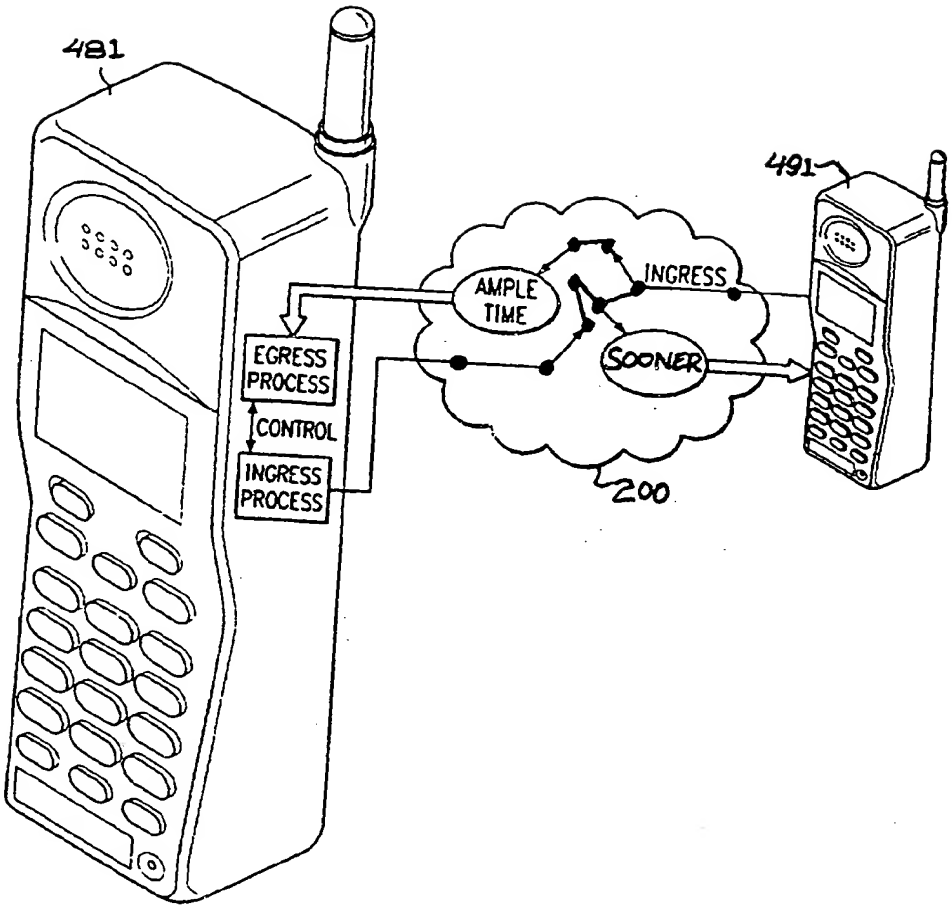

481
491
AMPLE TIME
INGRESS
SOONER
EGRESS PROCESS
CONTROL
INGRESS PROCESS
200

FIG. 4

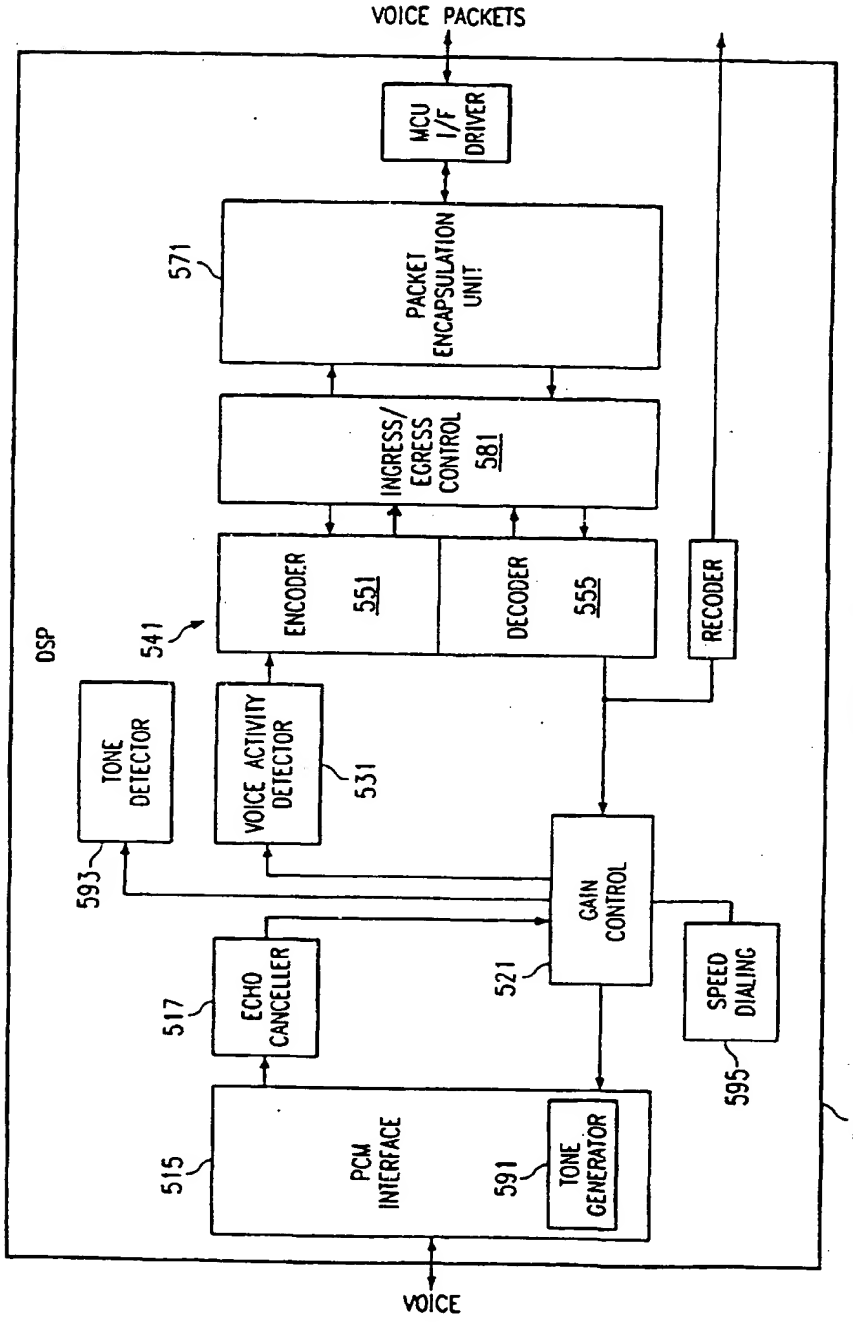

VOICE PACKETS
MCU I/F DRIVER
571
PACKET ENCAPSULATION UNIT
INGRESS/ EGRESS CONTROL 581
ENCODER 551
DECODER 555
RECODER
541
DSP
TONE DETECTOR
VOICE ACTIVITY DETECTOR
531
593
GAIN CONTROL
521
ECHO CANCELLER
517
SPEED DIALING
595
PCM INTERFACE
515
591
TONE GENERATOR
511
VOICE

FIG. 5

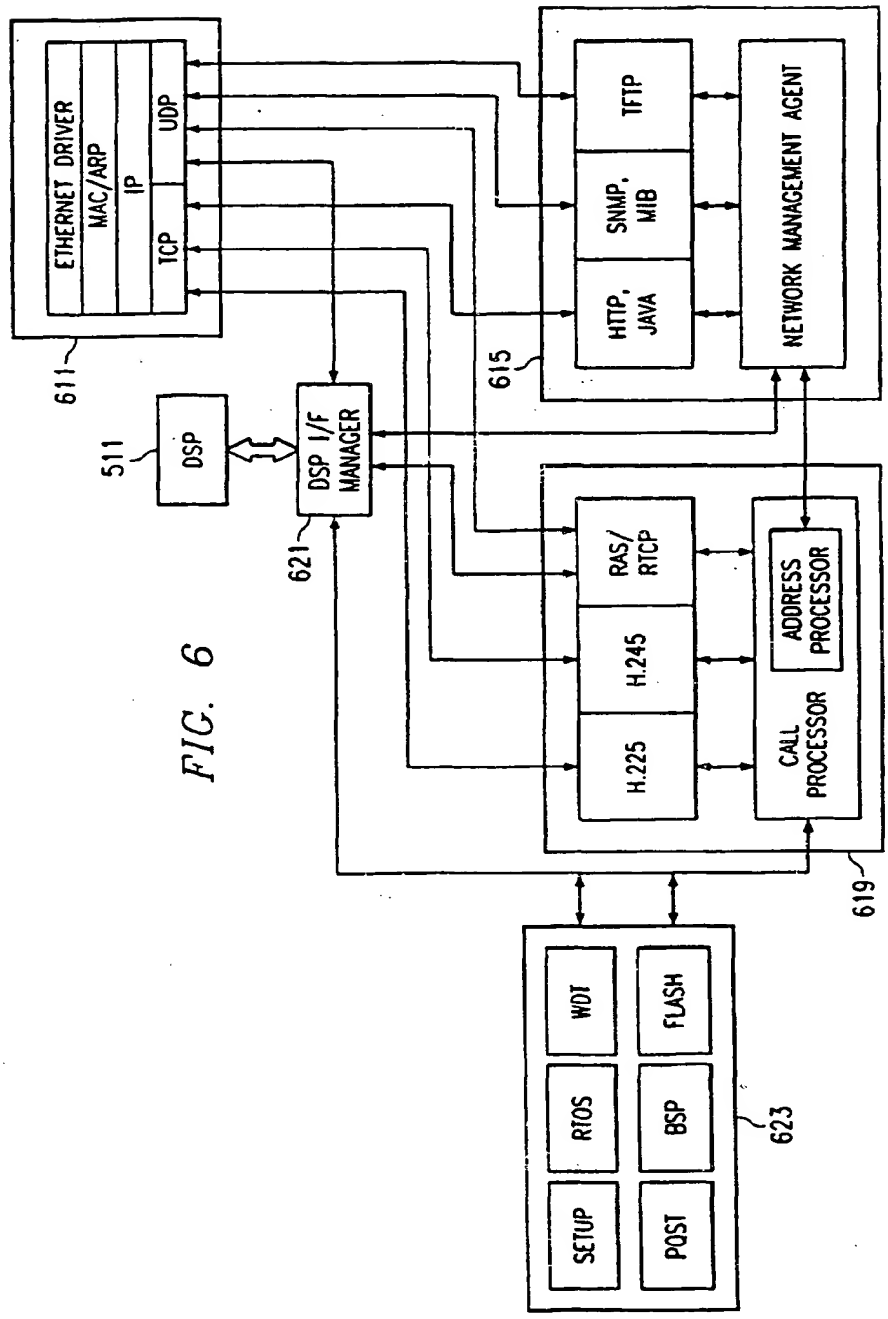

ETHERNET DRIVER
MAC/ARP
IP
TCP
UDP
611
DSP
511
DSP I/F MANAGER
621
TFTP
SNMP, MIB
HTTP, JAVA
615
NETWORK MANAGEMENT AGENT
RAS/ RTCP
H.245
H.225
ADDRESS PROCESSOR
CALL PROCESSOR
619
WDT
FLASH
RTOS
BSP
SETUP
POST
623

FIG. 6

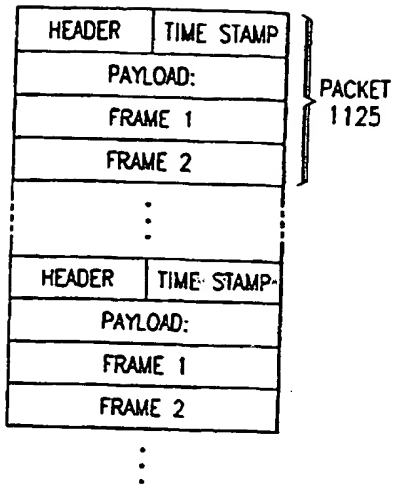

HEADER
TIME STAMP
PAYLOAD:
FRAME 1
FRAME 2
PACKET
1125
HEADER
TIME STAMP
PAYLOAD:
FRAME 1
FRAME 2

FIG. 7

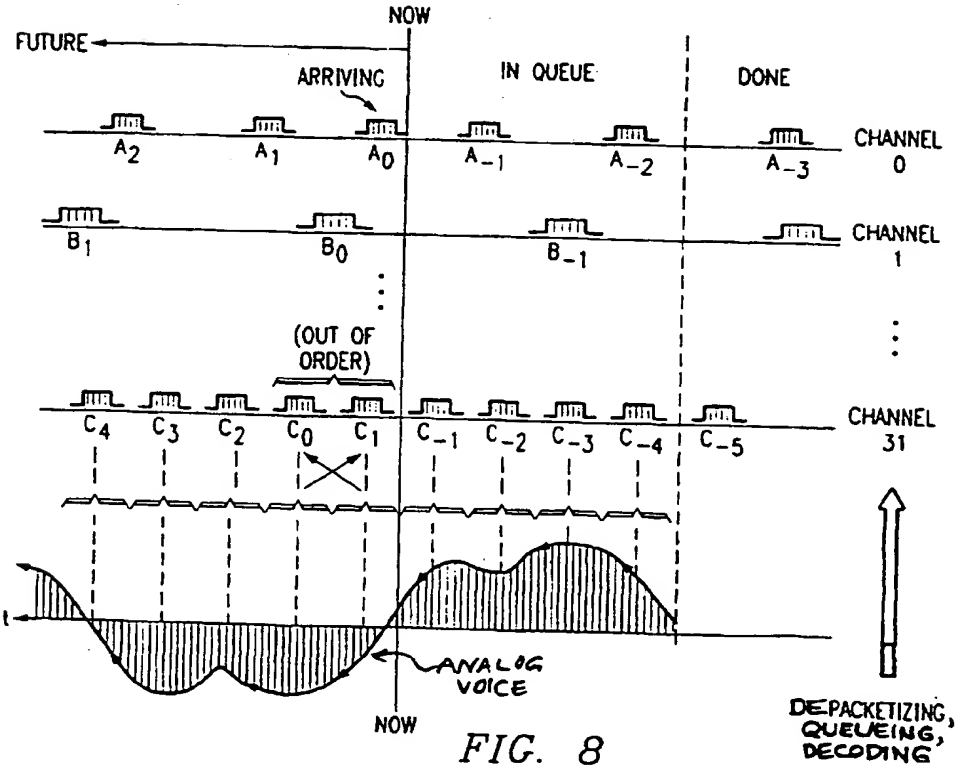

NOW
FUTURE
ARRIVING
IN QUEUE
DONE
CHANNEL 0
CHANNEL 1
CHANNEL 31
(OUT OF ORDER)
ANALOG VOICE
NOW
DEPACKETIZING, QUEUEING, DECODING

FIG. 8

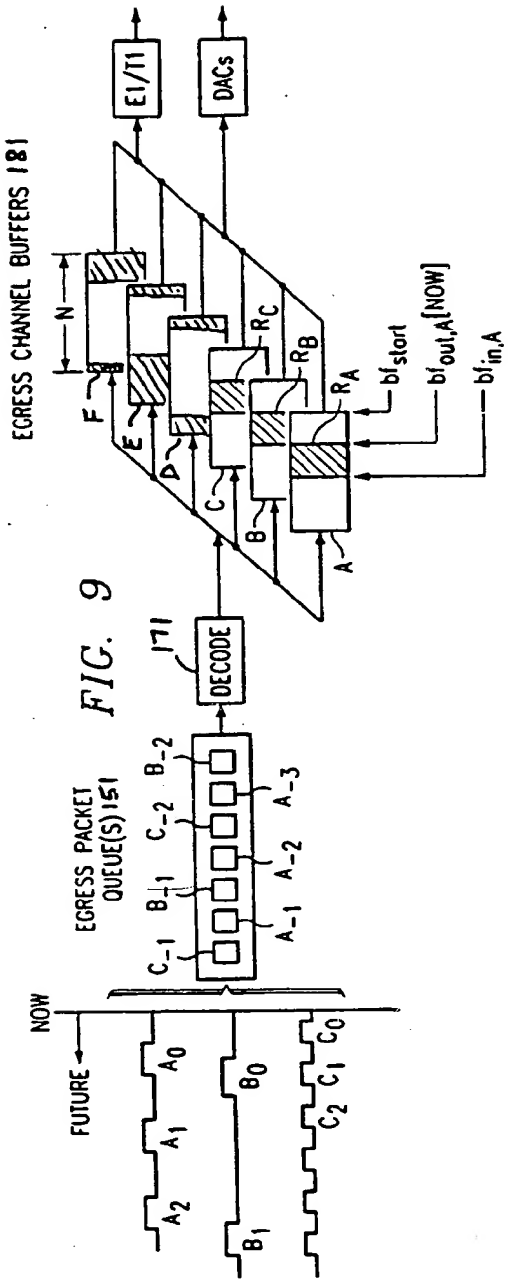

EGRESS CHANNEL BUFFERS 181
E1/T1
DACs
N
F
E
D
C
B
A
R_C
R_B
R_A
bf_{start}
$bf_{out,A}[NOW]$
$bf_{in,A}$
DECODE
171
EGRESS PACKET QUEUE(S) 151
C_{-1}
B_{-1}
A_{-1}
C_{-2}
B_{-2}
A_{-2}
A_{-3}
NOW
FUTURE
A_0
A_1
A_2
B_0
B_1
C_0
C_1
C_2

FIG. 9

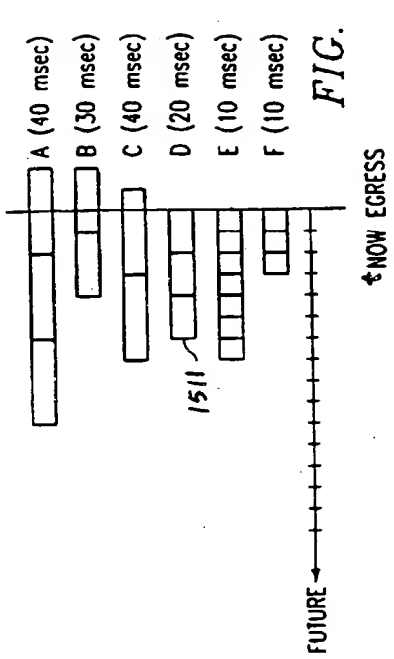

A (40 msec)
B (30 msec)
C (40 msec)
D (20 msec)
E (10 msec)
F (10 msec)
1511
NOW EGRESS
FUTURE

FIG. 10

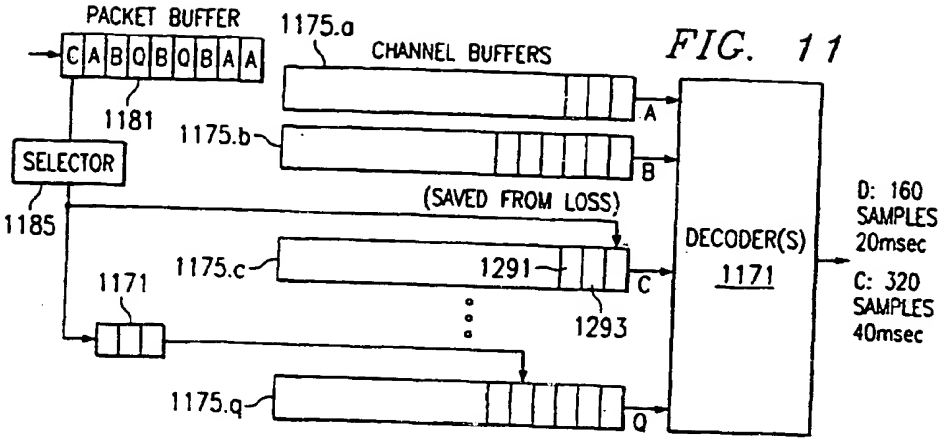


FIG. 11

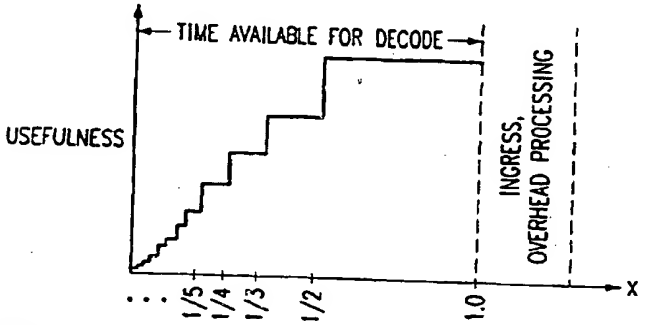


FIG. 12

$$x = \frac{\text{DECODE EXECUTE TIME}}{\text{TIME AVAILABLE FOR DECODE}}$$

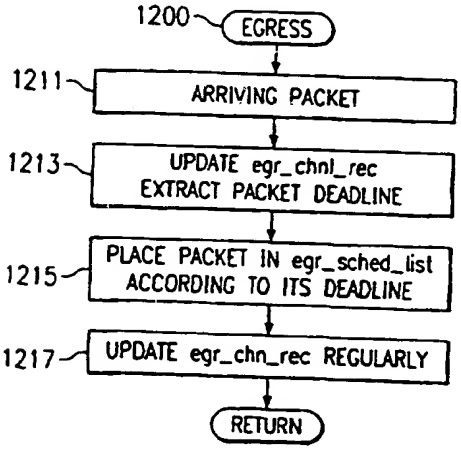


FIG. 13

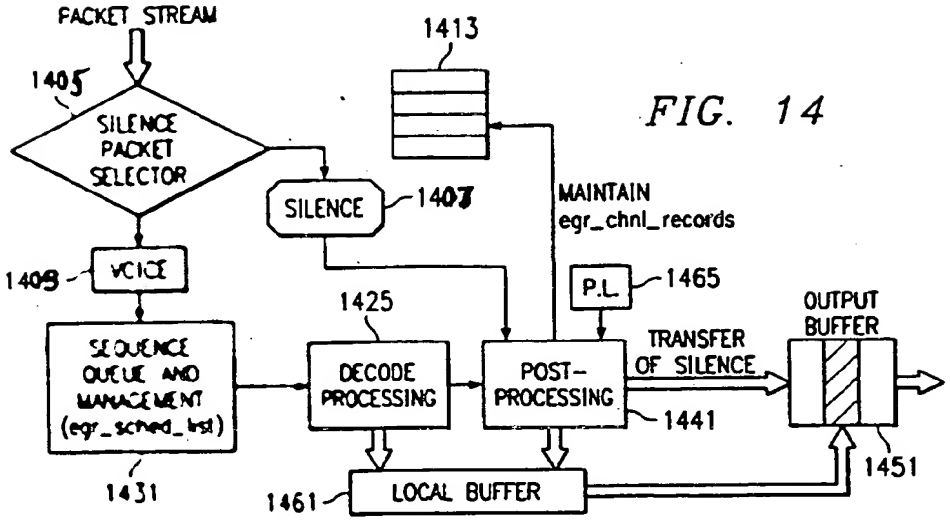

PACKET STREAM
1405
SILENCE PACKET SELECTOR
1413
SILENCE
1407
MAINTAIN egr_chnl_records
1409
VOICE
P.L.
1465
1425
OUTPUT BUFFER
SEQUENCE QUEUE AND MANAGEMENT (egr_sched_list)
DECODE PROCESSING
POST-PROCESSING
TRANSFER OF SILENCE
1441
1451
1431
1461
LOCAL BUFFER

FIG. 14

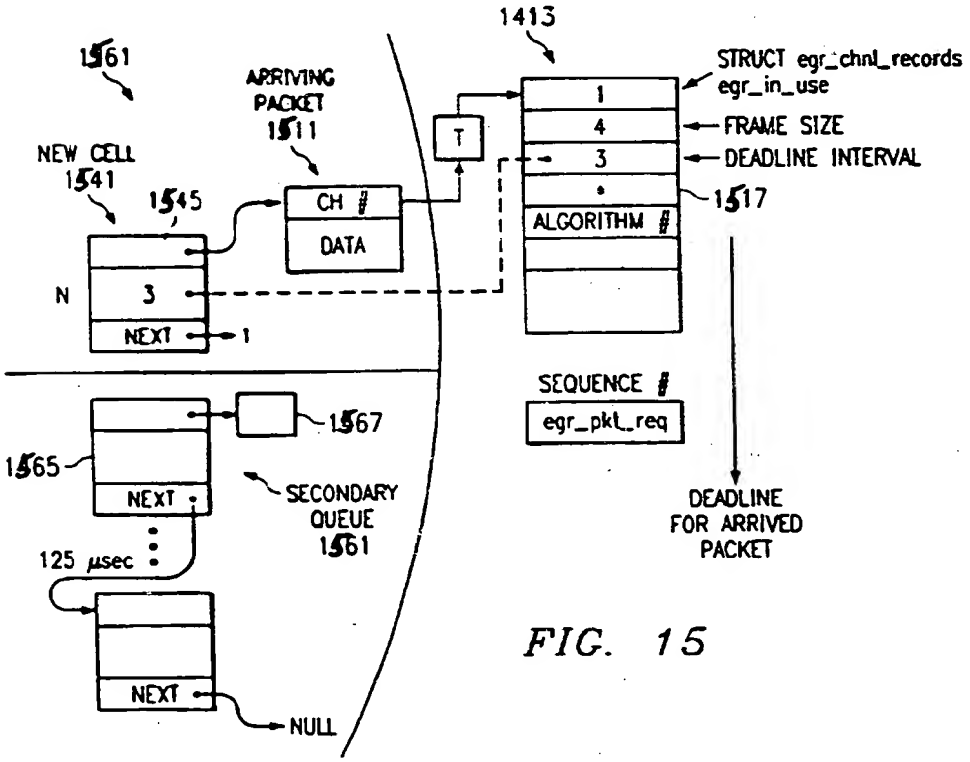

1413
1561
ARRIVING PACKET 1511
STRUCT egr_chnl_records
egr_in_use
1
4
FRAME SIZE
3
DEADLINE INTERVAL
1517
ALGORITHM #
NEW CELL 1541
1545
CH #
DATA
T
N
3
NEXT
1
SEQUENCE #
egr_pkt_req
1567
1565
NEXT
SECONDARY QUEUE 1561
125 μsec
DEADLINE FOR ARRIVED PACKET
NEXT
NULL

FIG. 15

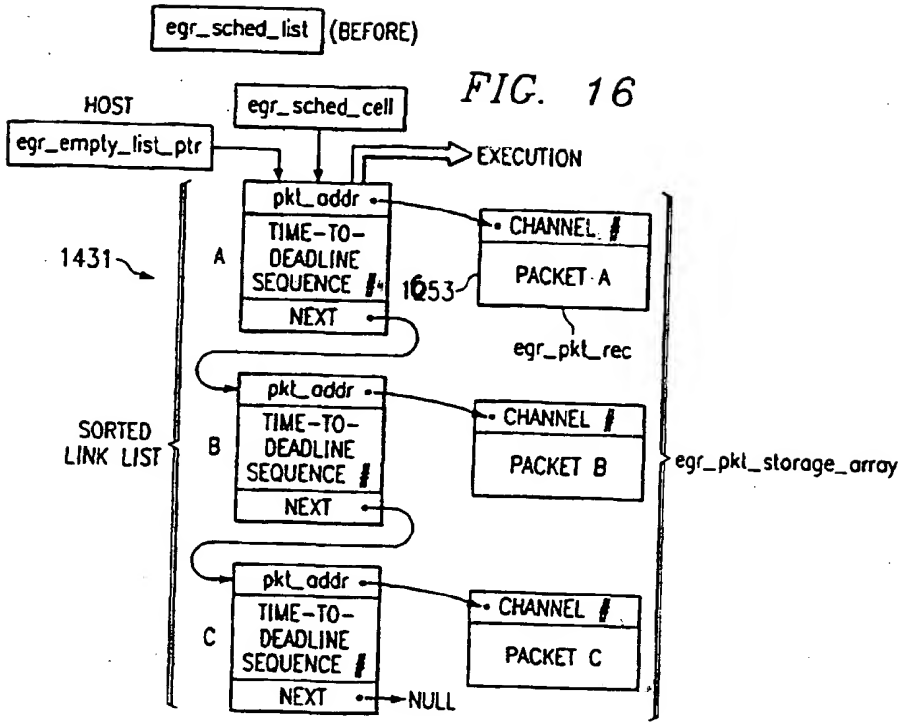


FIG. 16

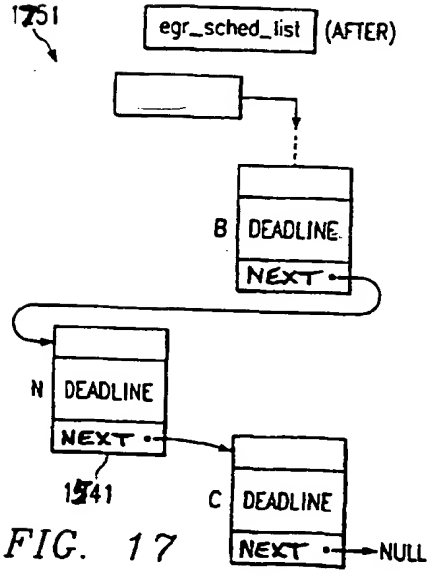


FIG. 17

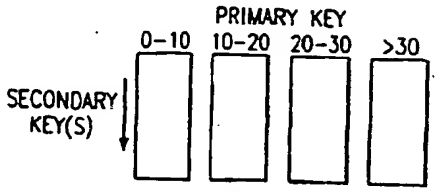


FIG. 18

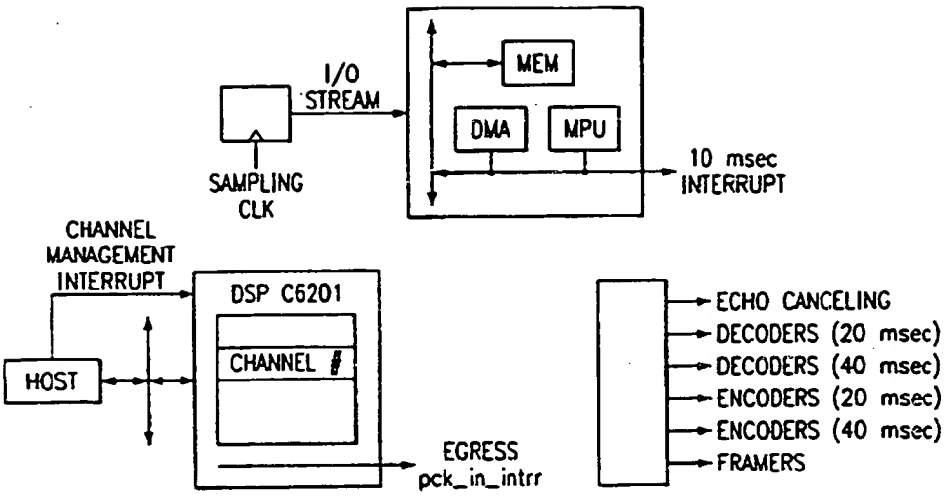

I/O
STREAM
SAMPLING
CLK
MEM
DMA
MPU
10 msec
INTERRUPT
CHANNEL
MANAGEMENT
INTERRUPT
DSP C6201
CHANNEL #
HOST
EGRESS
pck_in_intrr
ECHO CANCELING
DECODERS (20 msec)
DECODERS (40 msec)
ENCODERS (20 msec)
ENCODERS (40 msec)
FRAMERS

FIG. 19

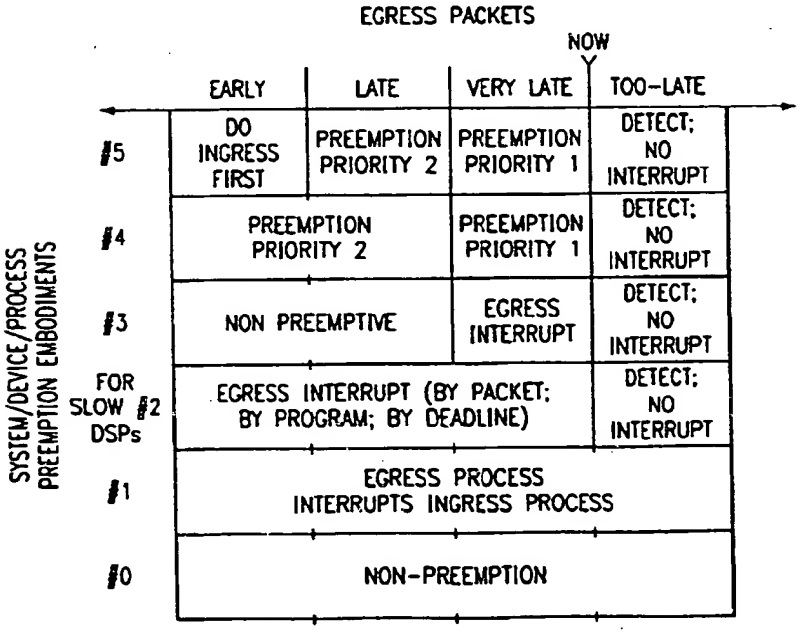

EGRESS PACKETS
NOW
EARLY
LATE
VERY LATE
TOO-LATE
SYSTEM/DEVICE/PROCESS PREEMPTION EMBODIMENTS
#5
DO INGRESS FIRST
PREEMPTION PRIORITY 2
PREEMPTION PRIORITY 1
DETECT; NO INTERRUPT
#4
PREEMPTION PRIORITY 2
PREEMPTION PRIORITY 1
DETECT; NO INTERRUPT
#3
NON PREEMPTIVE
EGRESS INTERRUPT
DETECT; NO INTERRUPT
FOR SLOW #2 DSPs
EGRESS INTERRUPT (BY PACKET; BY PROGRAM; BY DEADLINE)
DETECT; NO INTERRUPT
#1
EGRESS PROCESS INTERRUPTS INGRESS PROCESS
#0
NON-PREEMPTION

FIG. 20

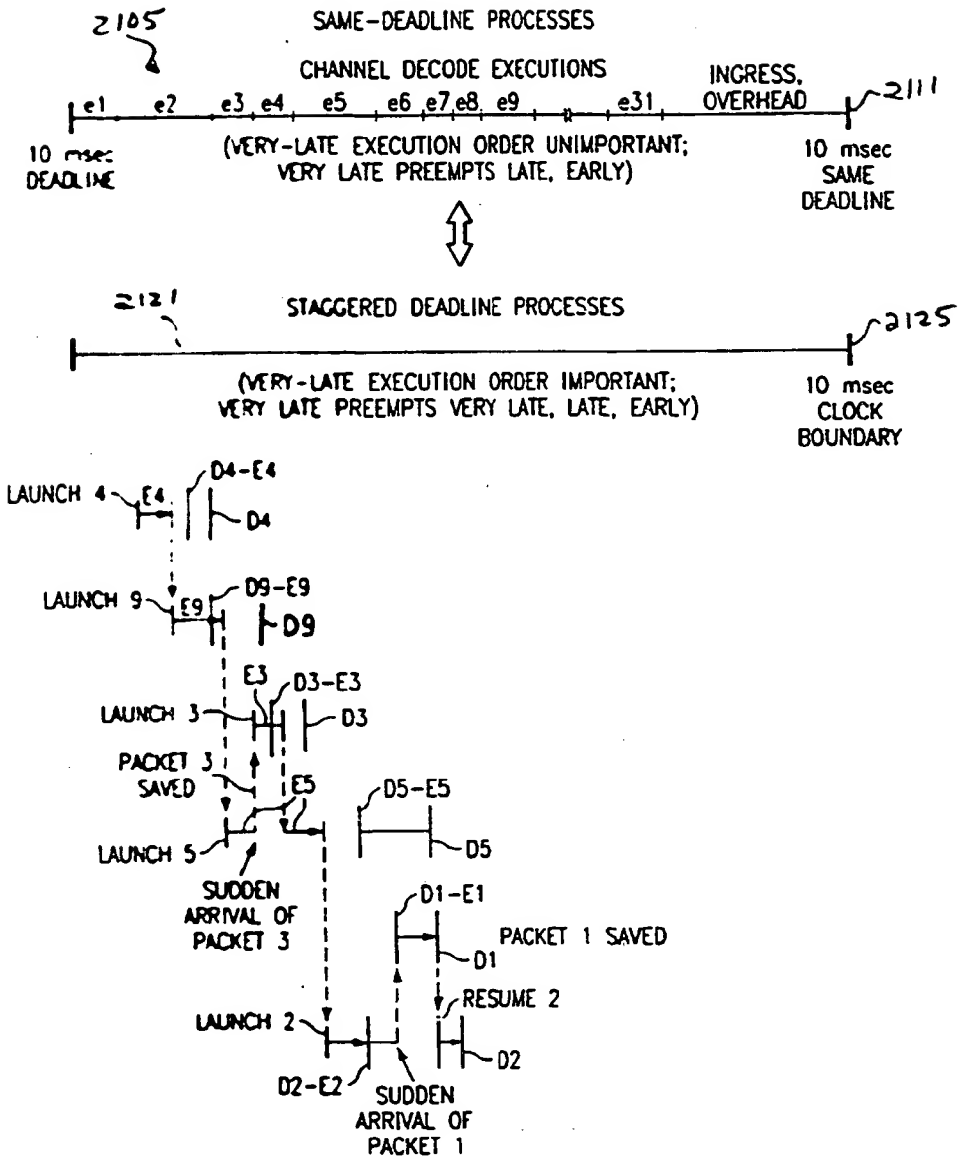

2105
SAME-DEADLINE PROCESSES
CHANNEL DECODE EXECUTIONS
e1 e2 e3 e4 e5 e6 e7 e8 e9 e31
INGRESS OVERHEAD
2111
10 msec DEADLINE
(VERY-LATE EXECUTION ORDER UNIMPORTANT; VERY LATE PREEMPTS LATE, EARLY)
10 msec SAME DEADLINE
2121
STAGGERED DEADLINE PROCESSES
2125
(VERY-LATE EXECUTION ORDER IMPORTANT; VERY LATE PREEMPTS VERY LATE, LATE, EARLY)
10 msec CLOCK BOUNDARY
LAUNCH 4
E4
D4-E4
D4
LAUNCH 9
E9
D9-E9
D9
E3
D3-E3
LAUNCH 3
D3
PACKET 3 SAVED
E5
D5-E5
LAUNCH 5
D5
SUDDEN ARRIVAL OF PACKET 3
D1-E1
PACKET 1 SAVED
D1
RESUME 2
LAUNCH 2
D2
D2-E2
SUDDEN ARRIVAL OF PACKET 1

FIG. 21

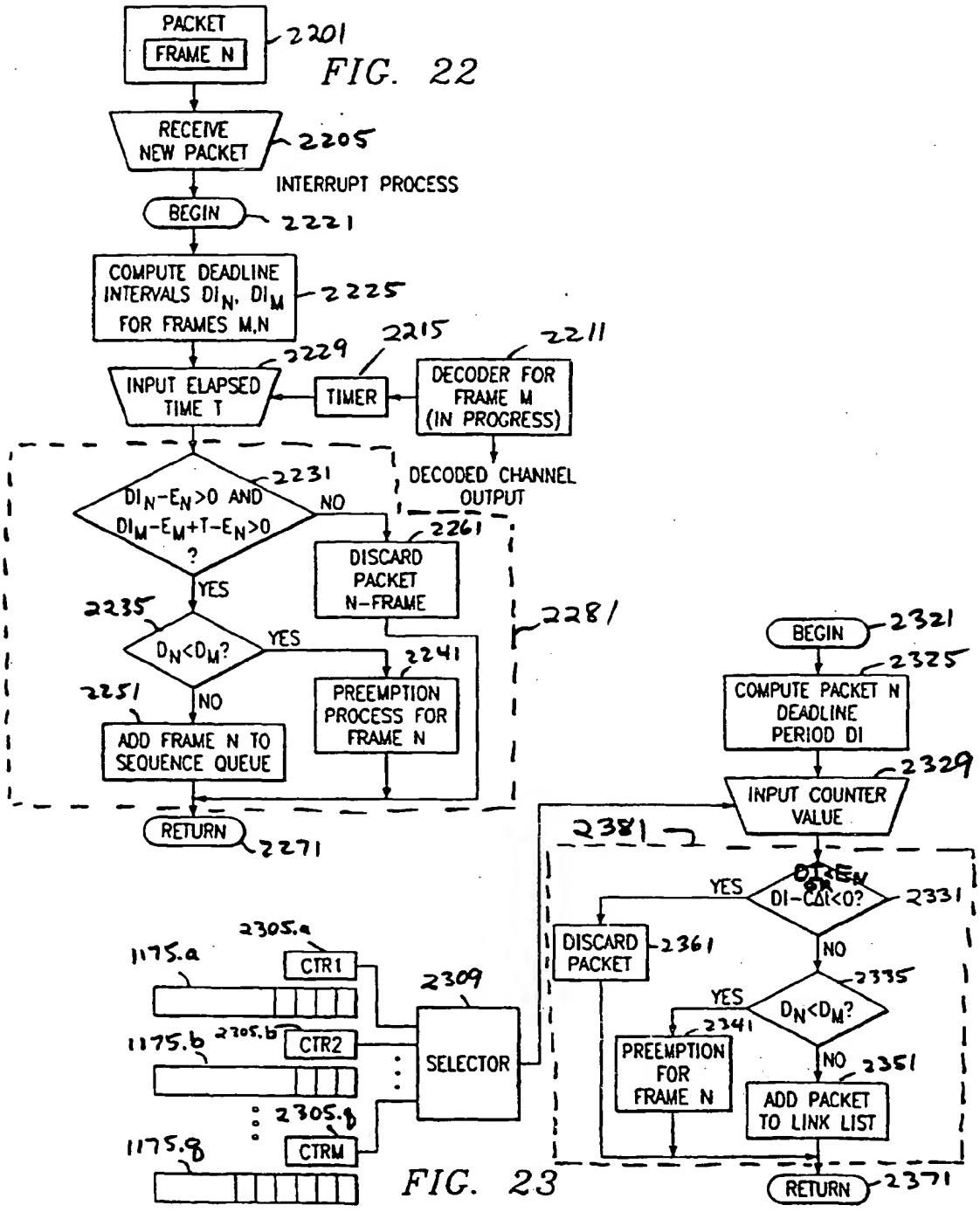

PACKET
FRAME N
2201
RECEIVE NEW PACKET
2205
INTERRUPT PROCESS
BEGIN
2221
COMPUTE DEADLINE INTERVALS DI_N, DI_M FOR FRAMES M,N
2225
INPUT ELAPSED TIME T
2229
TIMER
2215
DECODER FOR FRAME M (IN PROGRESS)
2211
DECODED CHANNEL OUTPUT
$DI_N-E_N>0$ AND $DI_M-E_M+T-E_N>0$?
2231
NO
YES
DISCARD PACKET N-FRAME
2261
$D_N<D_M$?
2235
YES
NO
2241
PREEMPTION PROCESS FOR FRAME N
2251
ADD FRAME N TO SEQUENCE QUEUE
2281
RETURN
2271
BEGIN
2321
COMPUTE PACKET N DEADLINE PERIOD DI
2325
INPUT COUNTER VALUE
2329
2381
$DI \geq E_N$?
YES
$DI-C\Delta I<0$?
2331
DISCARD PACKET
2361
NO
$D_N<D_M$?
2335
YES
NO
PREEMPTION FOR FRAME N
2341
ADD PACKET TO LINK LIST
2351
RETURN
2371
2305.a
CTR1
1175.a
2309
SELECTOR
2305.b
CTR2
1175.b
2305.n
CTRM
1175.n

FIG. 22

FIG. 23

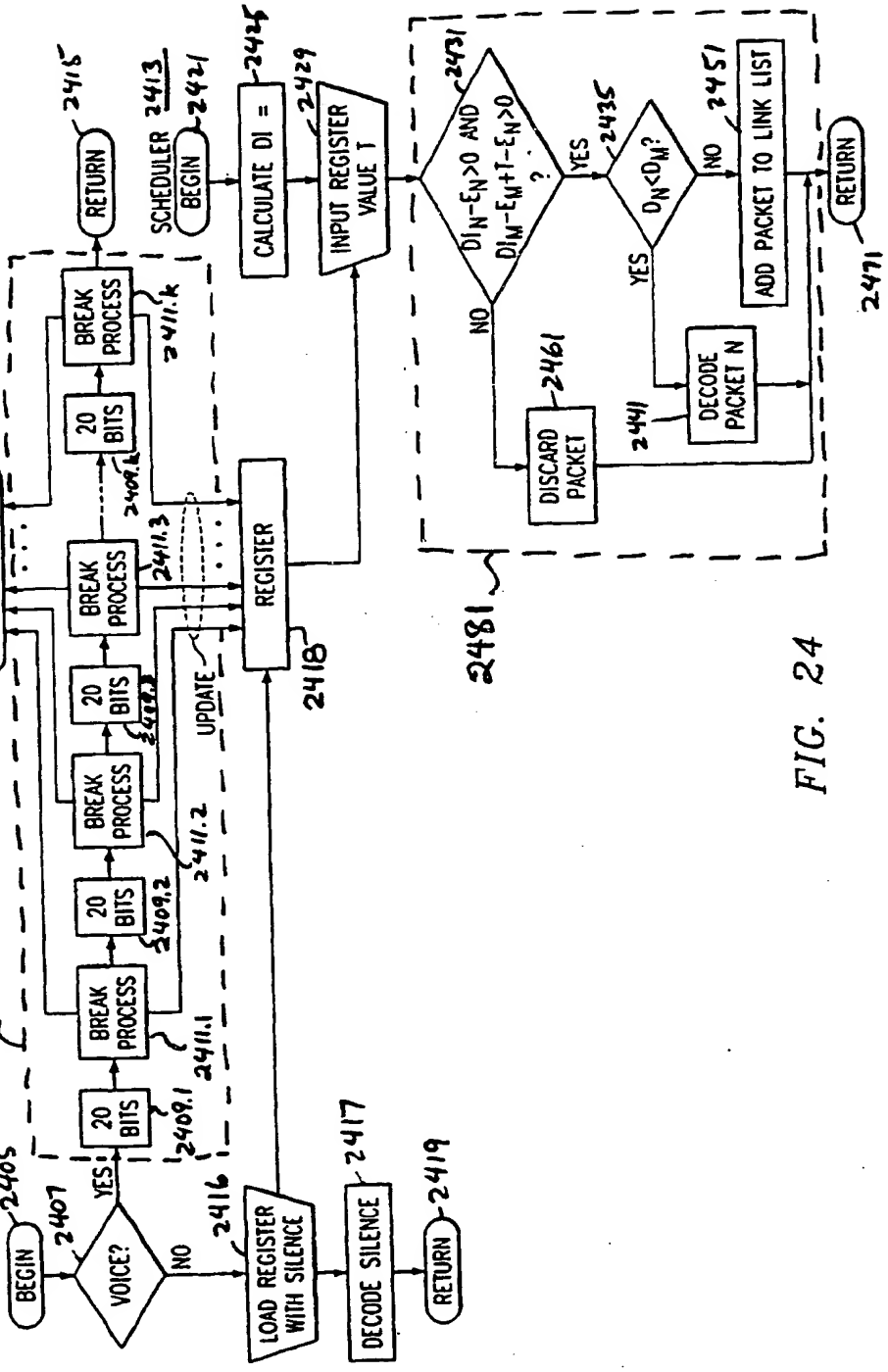

BEGIN
2405
2407
VOICE?
YES
NO
20 BITS
2409.1
BREAK PROCESS
2411.1
20 BITS
2409.2
BREAK PROCESS
2411.2
20 BITS
2409.3
BREAK PROCESS
2411.3
20 BITS
2409.k
BREAK PROCESS
2411.k
RETURN
2415
2403
DECODE PROCESS
CALL SCHEDULER
UPDATE
REGISTER
2418
2416
LOAD REGISTER WITH SILENCE
DECODE SILENCE
2417
RETURN
2419
SCHEDULER 2413
BEGIN
2421
CALCULATE DI =
2425
INPUT REGISTER VALUE T
2429
2481
$DI_N-E_N>0$ AND $DI_M-E_M+T-E_N>0$?
2431
NO
YES
DISCARD PACKET
2461
2441
$D_N<D_M$?
YES
NO
DECODE PACKET N
ADD PACKET TO LINK LIST
2451
RETURN
2471

FIG. 24

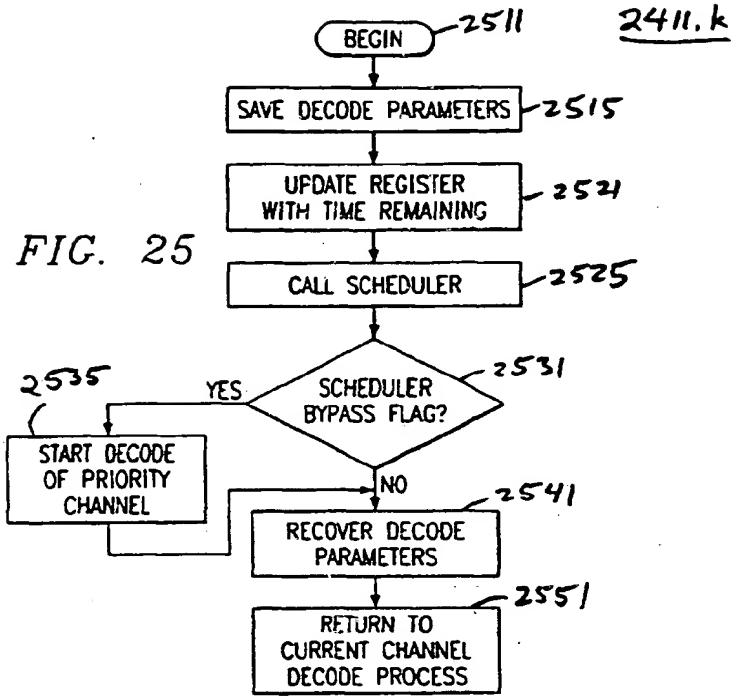

2411.k
BEGIN
2511
SAVE DECODE PARAMETERS
2515
UPDATE REGISTER WITH TIME REMAINING
2521
CALL SCHEDULER
2525
SCHEDULER BYPASS FLAG?
2531
YES
2535
START DECODE OF PRIORITY CHANNEL
NO
RECOVER DECODE PARAMETERS
2541
RETURN TO CURRENT CHANNEL DECODE PROCESS
2551

FIG. 25

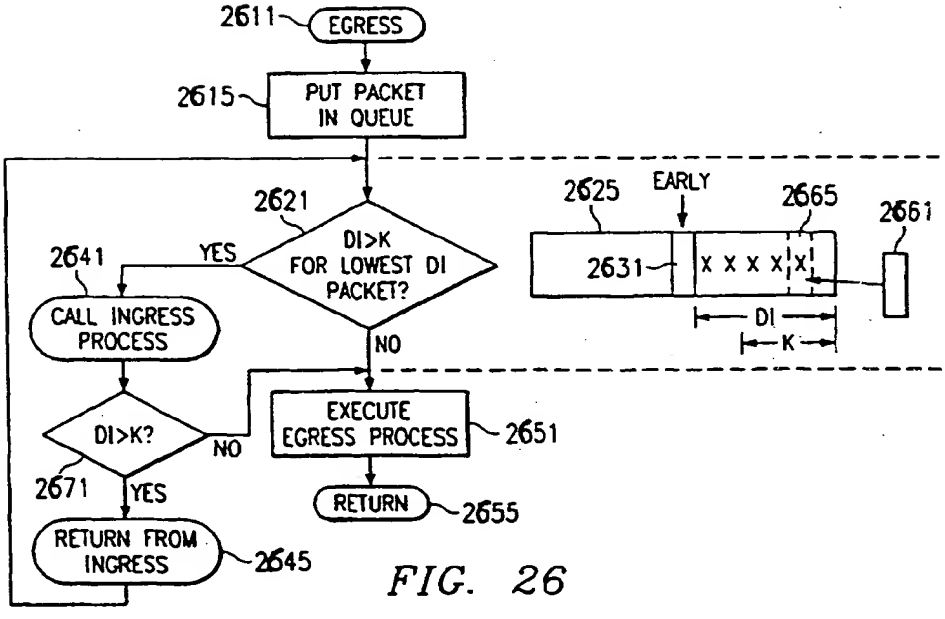

2611
EGRESS
2615
PUT PACKET IN QUEUE
2621
DI>K FOR LOWEST DI PACKET?
YES
2641
CALL INGRESS PROCESS
NO
DI>K?
NO
2671
YES
RETURN FROM INGRESS
2645
EXECUTE EGRESS PROCESS
2651
RETURN
2655
2625
EARLY
2665
2661
2631
X X X X X
DI
K

FIG. 26

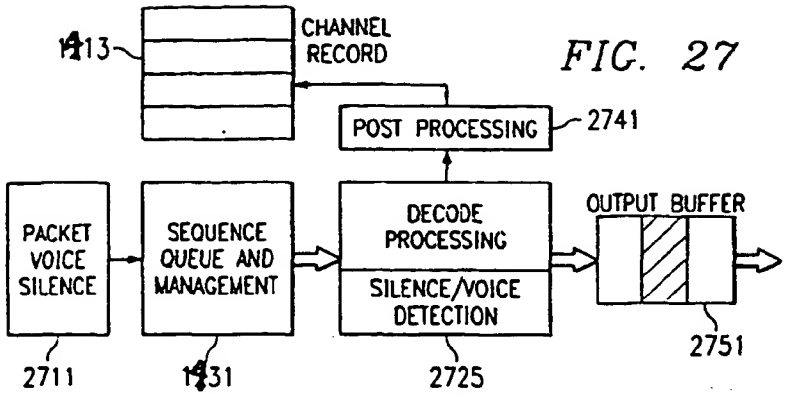

CHANNEL RECORD
2713
POST PROCESSING
2741
PACKET VOICE SILENCE
SEQUENCE QUEUE AND MANAGEMENT
DECODE PROCESSING
SILENCE/VOICE DETECTION
OUTPUT BUFFER
2751
2711
2731
2725

FIG. 27

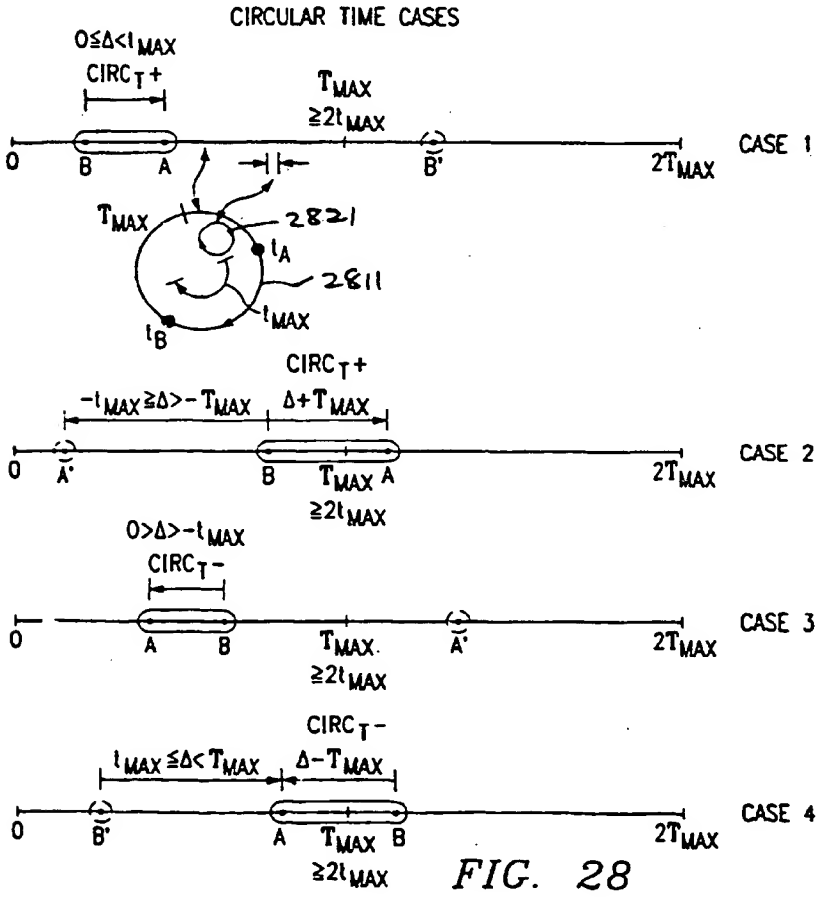

CIRCULAR TIME CASES
0≤Δ<t MAX
CIRC T+
T MAX
≥2t MAX
CASE 1
2821
2811
CIRC T+
-t MAX≥Δ>-T MAX
Δ+T MAX
CASE 2
0>Δ>-t MAX
CIRC T-
CASE 3
CIRC T-
t MAX≤Δ<T MAX
Δ-T MAX
CASE 4

FIG. 28

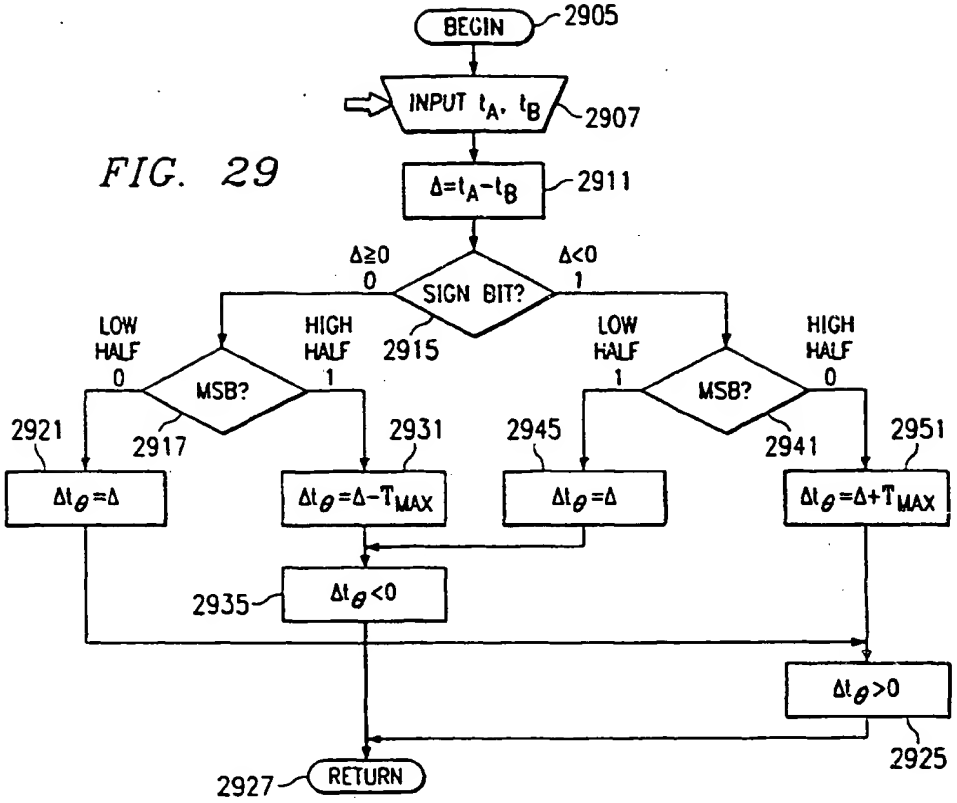

BEGIN
2905
INPUT t_A, t_B
2907
$\Delta=t_A-t_B$
2911
$\Delta\geq 0$
0
SIGN BIT?
$\Delta<0$
1
2915
LOW HALF
0
MSB?
HIGH HALF
1
2917
2921
2931
$\Delta t_\theta=\Delta$
$\Delta t_\theta=\Delta-T_{MAX}$
LOW HALF
1
MSB?
HIGH HALF
0
2941
2945
2951
$\Delta t_\theta=\Delta$
$\Delta t_\theta=\Delta+T_{MAX}$
2935
$\Delta t_\theta<0$
$\Delta t_\theta>0$
2925
2927
RETURN

FIG. 29

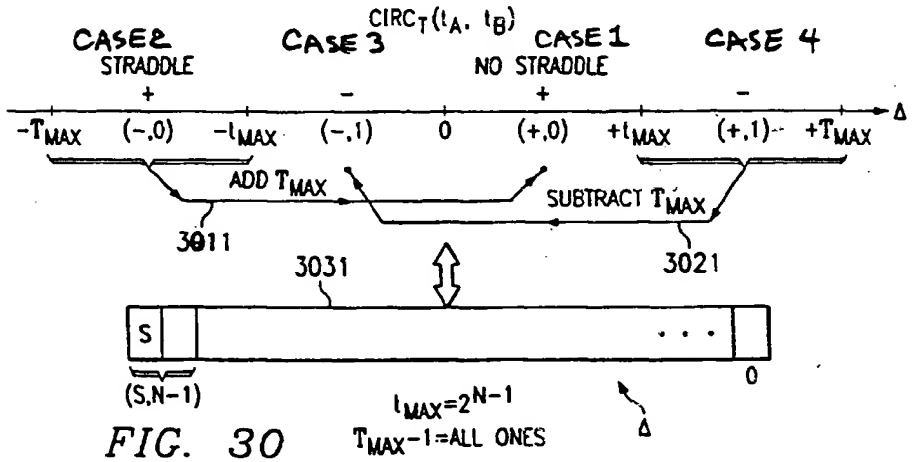

$CIRC_T(t_A, t_B)$
CASE 2
STRADDLE
CASE 3
CASE 1
NO STRADDLE
CASE 4
+
−
+
−
$-T_{MAX}$
(−,0)
$-t_{MAX}$
(−,1)
0
(+,0)
$+t_{MAX}$
(+,1)
$+T_{MAX}$
Δ
ADD T_{MAX}
SUBTRACT T_{MAX}
3011
3021
3031
S
0
(S,N−1)
$t_{MAX}=2^{N-1}$
$T_{MAX}-1$=ALL ONES
Δ

FIG. 30

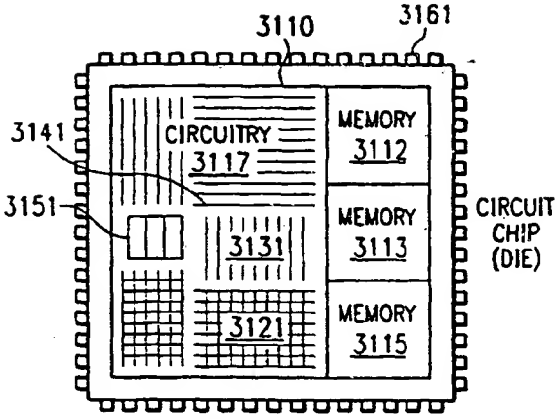

3110
3161
3141
3151
CIRCUITRY
3117
3131
3121
MEMORY
3112
MEMORY
3113
MEMORY
3115
CIRCUIT
CHIP
(DIE)

FIG. 31

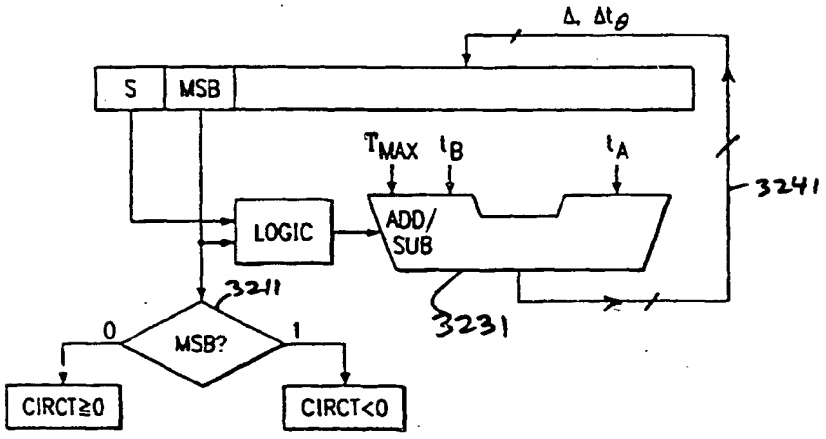

Δ, Δt_θ
S
MSB
T_MAX
t_B
t_A
3241
LOGIC
ADD/
SUB
3211
3231
0
MSB?
1
CIRCT≥0
CIRCT<0

FIG. 32

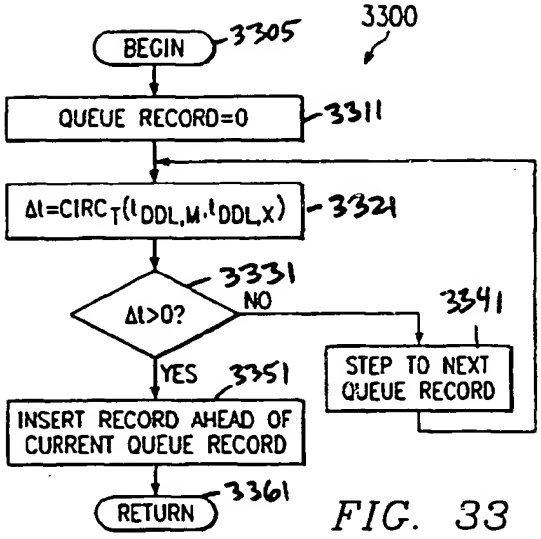


FIG. 33

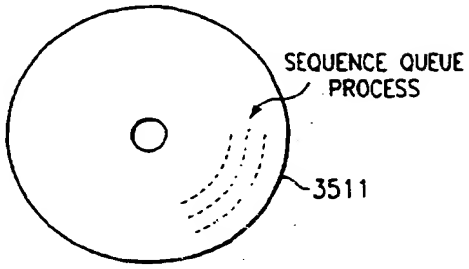


FIG. 35

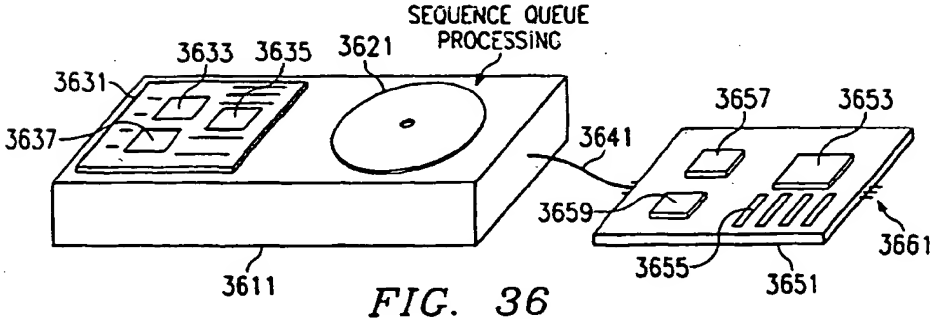


FIG. 36

FIG. 34

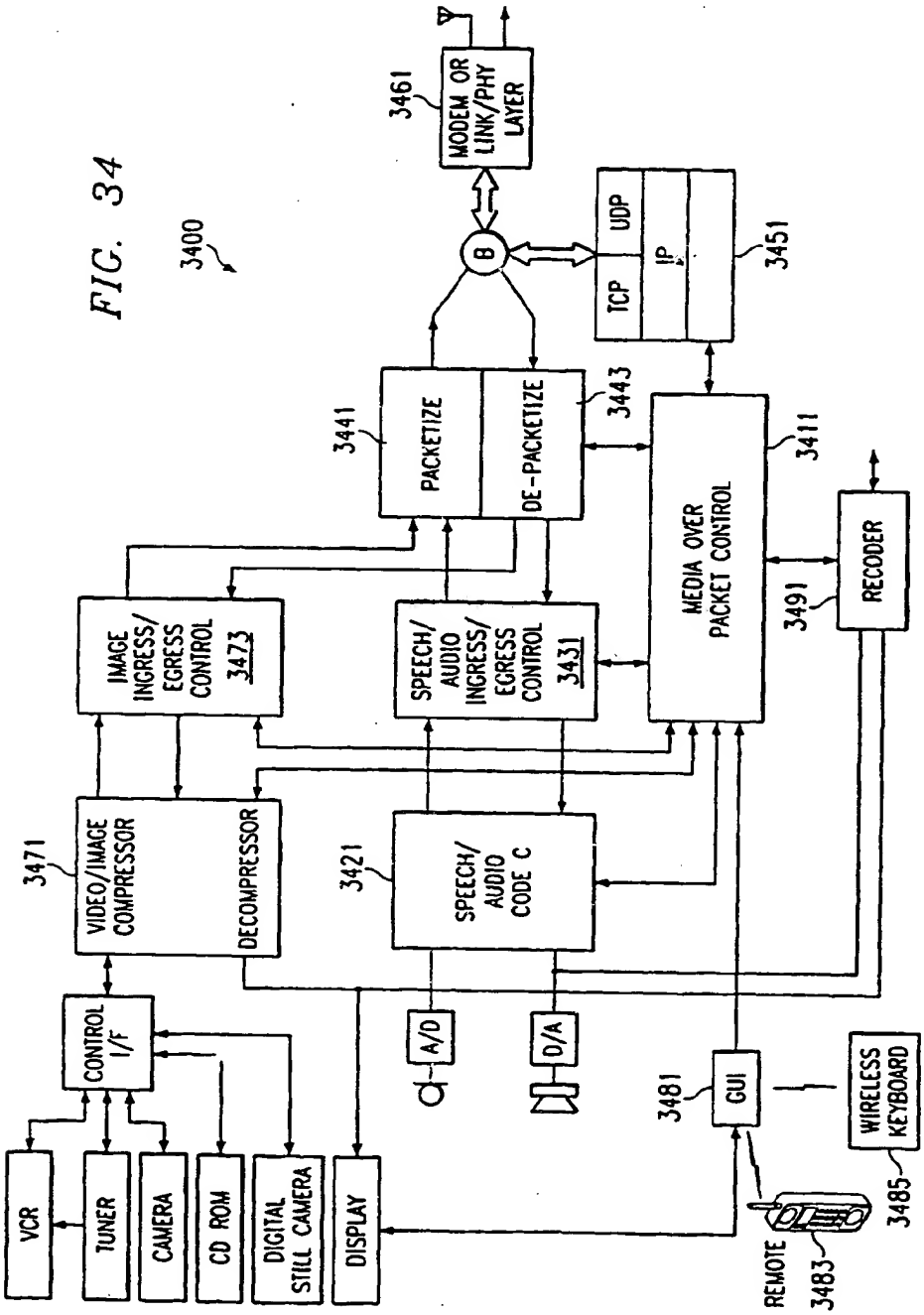
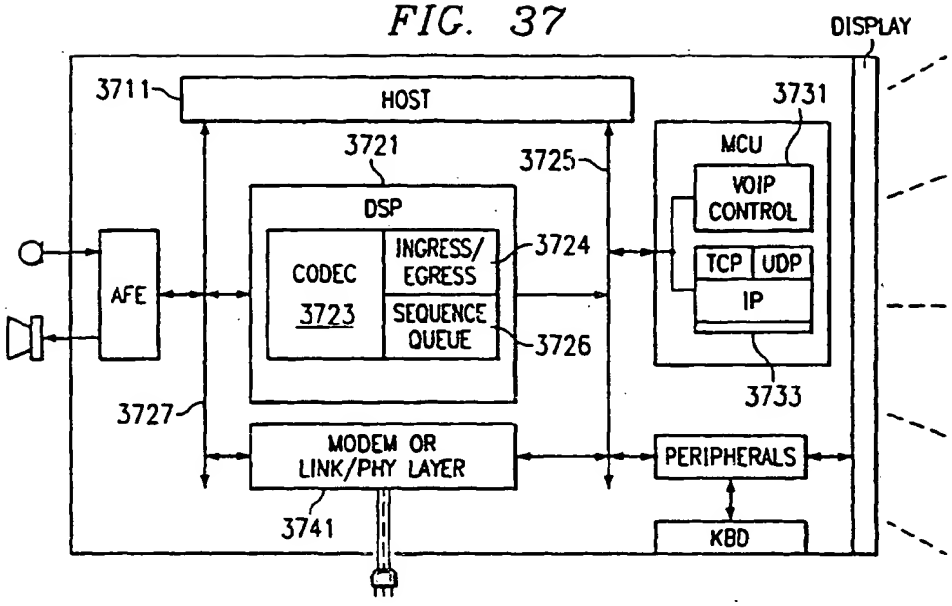

3400
3461
MODEM OR LINK/PHY LAYER
B
TCP
UDP
IP
3451
3441
PACKETIZE
DE-PACKETIZE
3443
3411
MEDIA OVER PACKET CONTROL
3491
RECODER
IMAGE INGRESS/ EGRESS CONTROL 3473
SPEECH/ AUDIO INGRESS/ EGRESS CONTROL 3431
3471
VIDEO/IMAGE COMPRESSOR
DECOMPRESSOR
3421
SPEECH/ AUDIO CODE C
CONTROL I/F
A/D
D/A
3481
GUI
WIRELESS KEYBOARD
3485
VCR
TUNER
CAMERA
CD ROM
DIGITAL STILL CAMERA
DISPLAY
REMOTE
3483

FIG. 37


DISPLAY
3711
HOST
3731
MCU
3721
3725
DSP
VOIP CONTROL
AFE
CODEC
3723
INGRESS/ EGRESS
3724
TCP
UDP
IP
SEQUENCE QUEUE
3726
3733
3727
MODEM OR LINK/PHY LAYER
PERIPHERALS
3741
KBD

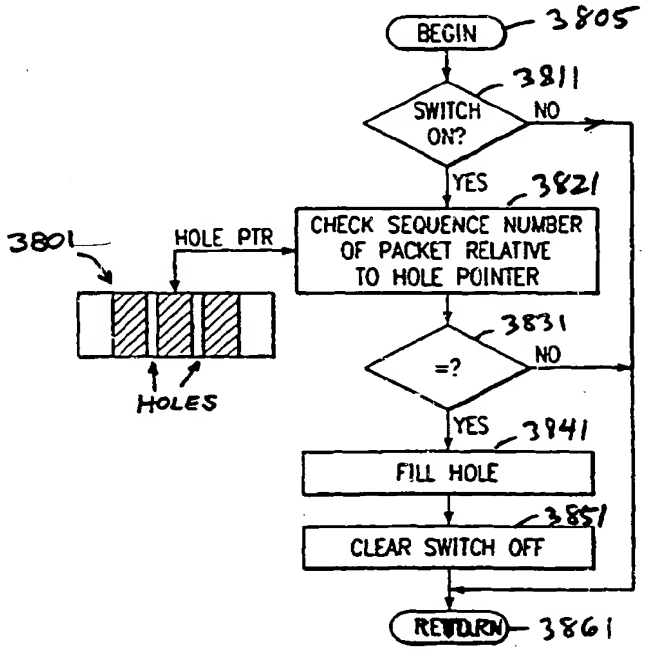

BEGIN
3805
3811
SWITCH ON?
NO
YES
3821
3801
HOLE PTR
CHECK SEQUENCE NUMBER OF PACKET RELATIVE TO HOLE POINTER
3831
=?
NO
HOLES
YES
3841
FILL HOLE
3851
CLEAR SWITCH OFF
RETURN
3861

FIG. 38